Julkaisujärjestelmän suunnittelu ja toteutus

Case Hämeen ammattikorkeakoulun kirjasto- ja tietopalvelut

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2013

Samu Lehtimäki

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Samu Lehtimäki **Vuosi** 2013

Työn nimi Julkaisujärjestelmän suunnittelu ja toteutus
Case Hämeen ammattikorkeakoulun kirjasto- ja tietopalvelut

TIIVISTELMÄ

Työn tavoitteena oli suunnitella ja toteuttaa julkaisujärjestelmä, joka tulisi korvaamaan käsin tehtävän virhealttiin ja hitaan tiedonkäsittelyprosessin. Työn toimeksiantajana toimi Hämeen ammattikorkeakoulun kirjasto- ja tietopalvelut, joka tarjoaa tietoaineistoja oppimisen, opetuksen sekä tutkimus- ja kehitystyön tukemiseen.

Hämeen ammattikorkeakoulussa tehtävien julkaisujen tiedot on tallennettu julkaisurekisteriin, joka on toteutettu osana Vanaicat-kokoelmatietokantaa. Sen ongelmana on, että vuodesta 2012 alkaen kaikkia opetus- ja kulttuuriministeriön vaatimia tietoja ei ole voitu tallentaa ilman, että se vaikeuttaisi oleellisesti kokoelmatietokannan ensisijaista peruskäyttöä.

Työssä toteutetun järjestelmän tarkoituksena oli mahdollistaa julkaisujen metatietojen lisääminen järjestelmään sekä opetus- ja kulttuuriministeriölle lähetettävien taulukkorakenteisien tiedostojen luominen suoraan järjestelmän tietokannasta oikeassa muodossa. Työssä syvennyttiin julkaisujärjestelmän toteutus- ja suunnitteluprosesseihin sekä käytiin läpi PHP-järjestelmien tietoturvauhkia. Lisäksi työssä tutustuttiin tietokantoihin ja niiden normalisointiin sekä työn toteuttamisessa käytettyihin tekniikoihin.

Tutkimusmenetelmänä työssä käytettiin konstruktiivista tutkimusta, eli tutkimusta jossa kehitetään ratkaisu työn tutkimusongelmiin. Tietolähteinä työssä käytettiin toimeksiantajan kanssa käytyjä keskusteluita ja opinnäytetyölle luotua Wikiä sekä PHP- ja MySQL-aiheisia kirjoja. Tämän lisäksi lähteinä käytettiin erilaisia verkkotekstejä ja opintomateriaalia.

Työn tuloksena syntyi toimiva ja tietoturvallinen runko tietokantoineen julkaisujärjestelmälle, joka tulee parantamaan pienellä jatkokehittämisellä tällä hetkellä käytössä olevaa tiedonkäsittelyprosessia. Toimeksiantajalle vaikeuksia aiheuttanut taulukkorakenteisen tiedoston luominen saatiin onnistuneesti toteutettua järjestelmään, kuten myös käyttäjätilien hallintaan sekä järjestelmän tietoturvaan liittyvät toiminnot.

Avainsanat tietoturva, tietokanta, ohjelmointi, PHP, MySQL

Sivut 46 s. + liitteet 5 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Visamäki
Degree Programme in Business Information Technology
System Engineering

Author Samu Lehtimäki **Year** 2013

Subject of Bachelor's thesis Publishing system design and implementation
Case HAMK University of Applied Sciences,
Library and Information Services

ABSTRACT

The purpose of this thesis commissioned by the Library and Information Services unit of HAMK University of Applied Sciences was to design and implement a publishing system, which would replace the error-prone manual activities and the slow data handling process.

Publications released by the HAMK University of Applied Sciences are stored in the publishing register of the Vanaicat collection database. The problem is that from 2012, all the information required by the Ministry of Education and Culture could not be inserted into the publishing register, without substantially complicating the primary basic operations of the database.

The purpose of the system implemented in this thesis was to enable the insertion of all required metadata and exporting data from the system to spreadsheets in a correct format. The implementation and design processes and the security threats of the PHP systems were examined in the thesis. In addition, databases and their normalization and the technologies used for implementation were introduced.

A constructive research method was used to develop a solution for the research problems identified in the thesis. The sources of information were discussions with the commissioner, Wiki section created for the thesis, PHP and MySQL-related publications. In addition, a lot of information was found in Internet and course materials.

The result of the thesis was a functional and secure frame with a database for the publishing system, which will with a little further development, improve the currently used data handling process. The creation of the spreadsheets and management of user accounts and system security functions were successfully implemented in the system.

Keywords information security, database, programming, PHP, MySQL

Pages 46 p. + appendices 5 p.

KÄSITELUETTELO

AJAX *(Asynchronous JavaScript and XML)*
Ryhmä web-sovelluskehityksessä käytettäviä tekniikoita, joilla web-sovelluksiin saadaan lisättyä vuorovaikutteista toiminnallisuutta ilman, että sivua täytyy päivittää.

Apache
Useille käyttöjärjestelmille saatavilla oleva HTTP-palvelinohjelma, joka mahdollistaa tiedostojen jakamisen HTTP-protokollan yli. Apache on maailman käytetyin palvelinohjelma.

BOM *(Byte order mark)*
Unicode-merkistön käyttämä tavujärjestystä määrittelevä merkki, jolla voidaan kertoa, millä koodaustavalla tieto on kirjoitettu.

CRUD *(Create, Read, Update, Delete)*
Lyhenne, joka muodostuu neljästä tietojen käsittelemiseen tarvittavasta funktiosta eli tiedon luomisesta, lukemisesta, päivittämisestä ja poistamisesta.

CSRF *(Cross-site request forgery)*
Web-sovelluksissa esiintyvä tietoturva-aukko, jota käyttämällä sovellukseen pystytään syöttämään haitallisia komentoja toiselta verkkosivustolta.

CSV *(Comma-Separated Values)*
Tiedostomuoto, joka esittää tiedon taulukkorakenteisessa muodossa. Nimensä mukaisesti eri tietokentät on erotettu tiedostossa pilkuilla ja rivinvaihdoilla.

MD5
Tiivistealgoritmi, jota käytetään useimmiten salasanojen suojaamiseen. MD5 tuottaa 128-bittisen tiivisteen, mikä yleensä käytetään heksakoodattuna 32 merkkiä pitkänä merkkijonona.

Metatieto
Tietosisältöä kuvailevaa ja määrittelevää tietoa, jonka tarkoituksena on tehostaa tietosisällön käyttöä.

Moduuli
Itsenäinen osa ohjelmaa tai järjestelmää, jolla on oma tehtävänsä. Ohjelma voi koostua monesta eri moduulista, jolloin ohjelmaa kutsutaan modulaariseksi ohjelmaksi

Palvelin
Tietokone, joka suorittaa sille määrättyä palvelinohjelmistoa tai tietojärjestelmää. Palvelin eli serveri voi toimia joko paikallisesti palvelinohjelmis-

ton kanssa samassa tietokoneessa tai tietoverkon välityksellä. Yhdellä palvelimella voi sijaita monta samanaikaisesti toimivaa palvelinohjelmistoa.

PDO *(PHP Data Objects)*
PHP-kirjasto, joka mahdollistaa PHP-sovellusten tietojen tallentaminen tietokantatyypistä riippumatta. Abstraktitason tietokantametodien ansiosta PDO sopii käytettäväksi melkein kaikkien tietokantaohjelmistojen kanssa.

Protokolla
Protokollalla tarkoitetaan tietoliikenteestä puhuttaessa yhteyskäytäntöä, jonka avulla määritellään yhteyden muodostamisessa käytettävät säännöt.

SHA-1 *(Secure Hash Algorithm)*
Yhdysvaltojen kansallisen turvallisuusvirasto NSA:n kehittämä tiivistealgoritmi, jota käytetään tiedon salaamisessa. SHA-1 tuottaa 160-bittisen tiivisteen, joka esitetään heksakoodattuna 40 merkkiä pitkänä merkkijonona. SHA-1:stä pidetään MD5 salausta turvallisempana ratkaisuna.

SQL *(Structured Query Language)*
SQL on standardoitu kyselykieli, jota suurin osa relaatiotietokannoista käyttää kyselykielenään.

UTF-8
Unicode-merkistön käyttämä koodaustapana, joka on hyvin yhteensopiva erilaisten tietojärjestelmien kanssa. UTF-8 mahdollistaa latinalaisten aakkosten käyttämisen järjestelmässä.

Webropol
Verkon välityksellä toimiva kyselyjärjestelmä, jolla voidaan rakentaa kyselylomakkeita ja käsitellä vastauksia. Webropol mahdollistaa vastauksien tallentamisen esimerkiksi Excel-tiedostoon.

Wiki
Käyttäjien muokattavissa oleva sivusto, joka perustuu tehokkaaseen tiedonvälitykseen ja yhteistyöhön käyttäjien välillä.

XSS *(Cross-site scripting)*
Web-sovelluksissa esiintyvä tietoturva-aukko, jota käyttämällä sovellukseen voidaan syöttää haitallista koodia käyttäjän nähtäville.

SISÄLLYS

1 JOHDANTO

Tässä työssä syvennyttiin PHP-pohjaisen järjestelmän suunnittelu- ja toteutusvaiheisiin. Työn teoriaosuudessa tutustuttiin työn suunnittelussa ja toteutuksessa käytettyihin tekniikoihin, relaatiotietokantojen keskeisiin käsitteisiin sekä kartoitettiin PHP-järjestelmien tietoturvauhkia. Työssä toteutetun järjestelmän suunnittelemisesta kertovassa osuudessa esiteltiin järjestelmältä vaaditut toiminnalliset ja laadulliset vaatimukset, sekä niiden pohjalta suunniteltu tietokanta. Työn toteutuksesta kertovassa osuudessa kerrottiin järjestelmän toiminnallisista ja tietoturvallisista ratkaisuista, sekä esiteltiin käyttöliittymää. Työssä toteutettu järjestelmä on vielä kehitysvaiheessa, joten jatkokehitys oli myös olennainen osa tätä työtä.

Työn toimeksiantajana toimi Hämeen ammattikorkeakoulun kirjasto- ja tietopalvelut. Kirjasto- ja tietopalvelut tarjoaa tietoaineistoja oppimisen, opetuksen sekä tutkimus- ja kehitystyön tukemiseen. Hämeen ammattikorkeakoulun kirjasto palvelee ensisijaisesti henkilökuntaa ja opiskelijoita, mutta myös ulkopuolisilla on oikeus kirjaston palveluihin. Kirjastossa on käytettävissä Nelli-portaali, jossa on saatavilla paljon lehtiä, kirjoja sekä aineistoa elektronisessa muodossa. (HAMK 2013.)

Hämeen ammattikorkeakoulussa tehtävien julkaisujen tiedot tallennetaan julkaisurekisteriin, joka on toteutettu osana kirjaston Vanaicat-kokoelmatietokantaa. Sen ongelmana on, että vuodesta 2012 alkaen kaikkia opetus- ja kulttuuriministeriön vaatimia tietoja ei ole voitu tallentaa ilman, että se vaikeuttaisi oleellisesti kokoelmatietokannan ensisijaista peruskäyttöä. Julkaisujen metatiedot on vuodesta 2012 lähtien ilmoitettu webropol-verkkolomakkeella, josta ne on tallennettu Excel-tiedostoon. Excel-tiedostosta julkaisujen metatiedot on siirretty käsin kahteen CSV-tiedostoon opetus- ja kulttuuriministeriön vaatimassa muodossa. CSV-tiedostojen luominen käsin on kuitenkin virhealtis ja ajallisesti hidas prosessi.

Työn tavoitteena oli tuottaa Hämeen ammattikorkeakoulun kirjasto- ja tietopalveluille julkaisujärjestelmä, joka korvaisi käsin tehtävän virhealttiin ja hitaan tiedonkäsittelyprosessin. Järjestelmän tarkoituksena oli mahdollistaa julkaisujen metatietojen lisääminen järjestelmään sekä opetus- ja kulttuuriministeriölle lähetettävien CSV-tiedostojen luominen suoraan järjestelmän tietokannasta oikeassa muodossa.

Tutkimuskysymykset, joihin tässä työssä haettiin vastausta, olivat miten suunnitellaan ja toteutetaan tietoturvallinen julkaisujärjestelmä ja miten suunnitellaan toimiva tietokantaratkaisu järjestelmälle. Lisäksi työssä vastattiin kysymykseen, miten toteutettavasta järjestelmästä saadaan vietyä tietoa ulos CSV-muodossa.

2 KÄYTETYT TEKNIIKAT

Tässä luvussa kerrotaan työssä toteutettavan julkaisujärjestelmän toteutuksessa ja suunnittelussa käytettävistä tekniikoista, jotka voidaan jaotella ohjelmointikieliin, ohjelmistoihin sekä sovelluskehyksiin. Luvussa kerrotaan myös, miten kyseisiä tekniikoita on käytetty hyväksi julkaisujärjestelmän suunnittelussa ja toteutuksessa.

Ohjelmistolla tarkoitetaan tiettyjen tehtävien suorittamiseen luotua tietokoneohjelmaa (Immonen 2002). Ohjelmistoja voidaan käyttää apuvälineenä sovelluskehityksessä tai niitä voidaan käyttää osana isompaa järjestelmää. Sovelluskehyksellä tarkoitetaan eräänlaista runkoa, jonka päälle ohjelmisto rakennetaan. Sovelluskehyksen käyttäminen mahdollistaa valmiiksi rakennettujen moduulien, funktioiden ja tyylimääritysten käyttämisen osana sovelluskehitystä.

2.1 HTML ja HTML5

HTML eli Hypertext Markup Language on verkkosivujen koodauksessa käytettävä standardoitu merkintäkieli, jonka koodi on elementeistä muodostuvaa rakenteista tekstiä. Elementit esitetään kulmasulkeilla merkityillä tunnisteilla eli tageilla. HTML tuottaa staattisen web-dokumentin eli sivun, joka näyttää tiedon niin kuin se on syötetty. Koska staattinen web-dokumentti ei sisällä toiminnallisuutta, se päivittyy vain, kun HTML-koodia päivitetään. (Murach & Harris 2010, 6.)

HTML5 on uusin versio HTML-kielestä, mutta sitä ei ole vielä standardisoitu, sillä sen kehitys on kesken. Keskeneräisyydestä huolimatta, kaikki suosituimmat selainohjelmat tukevat HTML5-kieltä. HTML5:n tarkoitus on tuoda HTML-kieleen toiminnallisuutta, mikä ennen vaati ulkoisten liitännäisten käyttämistä selaimessa. HTML5:sen avulla voidaan esimerkiksi lisätä animaatioita, sovelluksia, musiikkia ja elokuvia verkkosivulle. (W3Schools.)

Esimerkissä 1 on esitetty yksinkertaisen HTML-sivuston rakenne. Jotta selain voisi lukea HTML-sivun, täytyy tiedostopäätteenä käyttää html-päätettä. HTML-sivu aloitetaan html-elementillä, jonka jälkeen head-elementin sisällä esitetään sivuston tunnistetiedot. Pakollisia tietoja head-elementissä on vain otsikko, mutta elementissä voidaan sisällyttää esimerkiksi sivulla suoritettavat scriptit eli komentosarjat, metatietoja sivustosta ja CSS-tyyliohjeita. Seuraavaksi HTML-sivulle lisätään body-elementti, joka sisältää itse sivuston sisällön. Esimerkissä 1 sivustolle tulostetaan h1-elementillä pääotsikko, ja p-elementillä tekstikappale. Kuvasta 1 voidaan havaita, miltä esimerkin 1 koodi näyttää suoritettuna selaimessa.

```
<html>
 <head>
  <title>HTML Esimerkki</title>
 </head>
 <body>
```

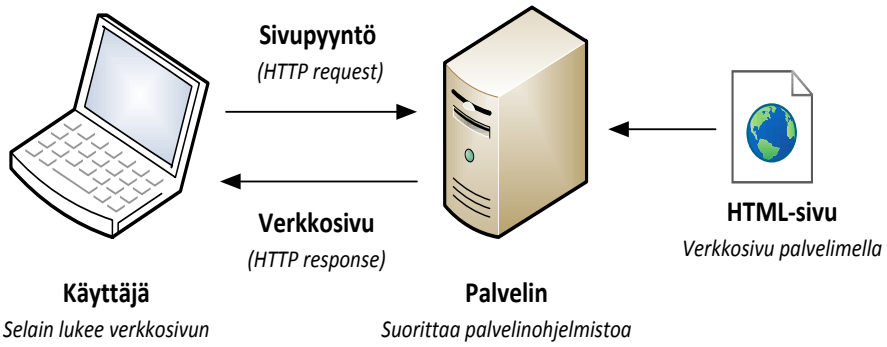
```
  <h1>HTML-sivu</h1>
  <p>Tähän voidaan kirjoittaa tekstiä.</p>
 </body>
</html>
```

Esimerkki 1. Yksinkertaisen HTML-sivuston rakenne

Kuva 1. Esimerkissä 1 esitetty koodi suoritettuna selaimessa

Protokollalla tarkoitetaan tietoliikenteestä puhuttaessa yhteyskäytäntöä, jonka avulla määritellään yhteyden muodostamisessa käytettävät säännöt. HTTP-protokollaa käytetään asiakkaan ja palvelimen väliseen tiedonsiirtoon, jolloin kommunikointi asiakkaan ja palvelimen välillä toteutuu asiakaspyyntöjen ja palvelinvastauksien muodossa. (Podila 2013.) HTTP-protokollan muodostamiseen voidaan käyttää myös suojattua yhteyttä, jolloin asiakaspyyntö ja palvelinvastaus salataan SSL-protokollan tai TLS-protokollan avulla.

Kuviossa 1 on esitetty staattisen HTML-dokumentin käsittelyprosessi. Käsittelyprosessi alkaa, kun käyttäjän selain lähettää palvelimelle sivupyynnön HTTP-protokollan avulla Palvelin etsii pyyntöä vastaavan sivun palvelimen levyasemalta ja palauttaa sen käyttäjän selaimelle. Selain suorittaa palautetun HTML-koodin ja näyttää pyydetyn sivun selaimessa. (Murach & Harris 2010, 6.)



Kuvio 1. Staattisen HMTL-dokumentin käsittelyprosessi

2.2 CSS ja CSS3

CSS eli Cascadin Style Sheets on tyyliohjekieli, jonka avulla web-dokumentille ehdotetaan säännöt, joita noudattaen dokumentti esitetään. Tyyliohjeita ovat mm. värit, fontit sekä sivuston ulkoasuun liittyvät muotoilut. Kun CSS-tyyliohjeita halutaan käyttää, lisätään tyyliohjeet suoraan HTML-koodiin, yleensä head-elementin sisälle. CSS-tyyliohjeet voidaan

lisätä joko viittaamalla erilliseen CSS-tiedostoon tai upottamalla tyyliohjeet suoraan HTML-dokumenttiin.

CSS3 on uusin standardi CSS-tyyliohjekielestä, mikä tukee kaikkia aikaisempia CSS-versioita. CSS3:sta kehitetään moduuleissa, joista jokaisesta vastaa oma kehitystiiminsä. Moduuleita ovat esimerkiksi taustakuvat ja reunukset, laatikot, tekstiefektit, animaatiot ja käyttöliittymät. CSS3:sella voidaan myös antaa selainkohtaisia muotoiluja etuliitteiden avulla. (Bookwalter 2011.)

Kuvassa 2 on esitetty kirjautumiseen tarkoitettu lomake, jossa ei käytetä ollenkaan CSS-tyyliohjeita. Kuvasta voidaan havaita, että asettelu noudattaa suoraan HTML-koodissa määriteltyä rakennetta, eikä sisällä esimerkiksi asetteluun, väreihin tai fontteihin vaikuttavia tyyliohjeita.

Kuva 2. Kirjautumiseen tarkoitettu lomake, joka ei sisällä CSS-tyyliohjeita

Esimerkissä 2 on esitetty CSS-tyyliohjeita. Tyyliohjeessa käytettävä #-merkki viittaa HTML-koodissa esitettyyn id-valitsimeen, jonka avulla elementeille voidaan antaa attribuutiksi eri arvoja ja täten erotella tyyliohjeet koskemaan vain haluttuja elementtejä. Elementin sisällä sijaitseville HTML-tunnisteille voidaan antaa tyyliohjeita lisäämällä tunnisteen nimi elementin nimen perään. Esimerkin 2 tapauksessa tyyliohjeita annetaan elementille, jonka attribuuttina HTML-koodissa on arvo ”login” sekä fieldset- ja input-tunnisteille kyseisen elementin sisällä.

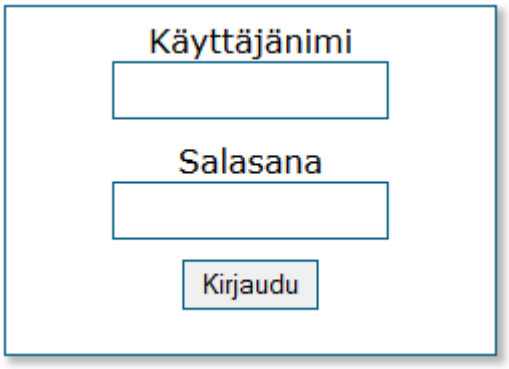
```
#login{
            margin-top:100px;
            margin-left:auto;
            margin-right:auto;
            text-align:center;
            width:250px;
            font-family:Verdana;
}

#login fieldset{
            border-style:solid;
            border-width:1px;
            border-color:#005a84;
            -moz-box-shadow:3px 3px 3px 1px #bababa;
            -webkit-box-shadow:3px 3px 3px 1px #bababa;
            box-shadow:3px 3px 3px 1px #bababa;
}
```

```
#login input{
            height:25px;
            border-style:solid;
            border-width:1px;
            border-color:#005a84;
            margin-bottom:10px;
}
```

Esimerkki 2. CSS- ja CSS3-tyyliohjeita

Kuvasta 3 voidaan havaita kirjautumiseen tarkoitettu lomake, jossa käytetään esimerkissä 2 esitettyjä CSS- ja CSS3-tyyliohjeita. Kuvan lomake on HTML-koodiltaan identtinen kuvassa 2 esitettyyn kirjautumislomakkeeseen, jotta CSS:n vaikutus visuaaliseen ulkoasuun voidaan havaita. CSS-tyyliohjeita on käytettyä lomakkeessa esimerkiksi kenttien ja reunojen väritykseen, tekstin muotoiluun sekä lomakkeen keskittämiseen elementin sisällä. CSS3:lla on toteutettu lomakkeen reunojen varjostukset.



Kuva 3. Kirjautumiseen tarkoitettu lomake, joka sisältää CSS ja CSS3-tyyliohjeita

2.3 JavaScript

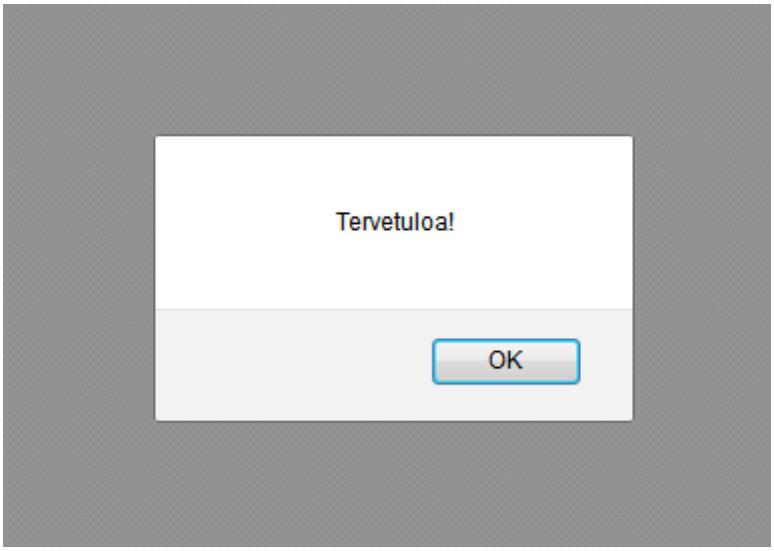
JavaScript on komentosarjakieli, jolla mahdollistetaan dynaamisen toiminnallisuuden lisääminen verkkosivuille. JavaScriptiä suoritetaan suoraan web-selaimessa ja tämä mahdollistaa esimerkiksi ponnahdusikkunoiden, tekstin, värin ja kuvien lisäämisen elementteihin web-sivulla. JavaScriptiä ei tule sekoittaa Java-ohjelmointikieleen. (Nixon 2012, 291.)

JavaScriptiä käytetään yleensä web-sovelluksissa jQuery-kirjaston muodossa. jQuery ei ole ohjelmointikieli, vaan hyvin kirjoitettu JavaScript-kirjasto, jolla sivustolle saadaan helposti lisättyä JavaScript koodia. Helppokäyttöisen ja joustavan rajapintansa ansiosta jQuery toimii selaimesta riippumatta ja sen avulla voidaan käydä helposti läpi HTML-dokumentteja ja manipuloida niitä, elävöittää toiminnallisuutta animaatioilla sekä tehdä AJAX:n käytöstä vaivatonta (The jQuery Foundation 2013). Koska kaikki toimii samalla tavalla selaimesta riippumatta, tekee jQuery eritoten tapahtumien käsittelemisestä vaivatonta (Falck 2008).

Esimerkissä 3 on esitetty komentosarja eli skripti, jolla jQuery saadaan käyttöön halutulle sivulle. Skripti lisätään useimmiten HTML:n head-elementin sisälle, mutta se voidaan lisätä myös ennen body-elementin sulkutagia. Esimerkissä on oletettu, että jQuery-kirjasto on tallennettu jquery.js -nimisenä tiedostona samaan kansioon, missä HTML-dokumentti sijaitsee. Kuvasta 4 voidaan havaita, miltä esimerkissä 3 esitetty komentosarja näyttää selaimessa suoritettuna. Kun HTML-dokumentti on ladattu, body-elementin sisälle kirjoitettu komentosarja tulostaa käyttäjälle sanan ”Tervetuloa!”.

```
<!DOCTYPE html>
<html>
 <head>
  <title>jQuery Esimerkki</title>
  <script type="text/javascript" src="jquery.js"></script>
 </head>
 <body>
  <script>$(function() { alert('Tervetuloa!') })</script>
 </body>
</html>
```

Esimerkki 3. jQueryn käyttöönotto HTML-koodissa



Kuva 4. Esimerkissä 3 esitetty komentosarja suoritettuna selaimessa

2.4 PHP

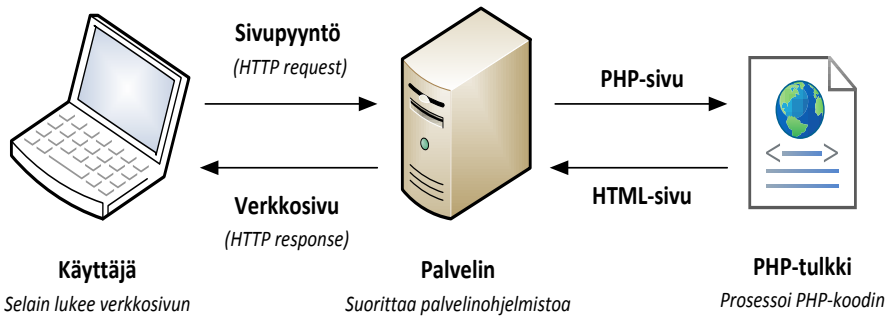
PHP eli PHP: Hypertext Preprocessor on Rasmus Lerdorffin vuonna 1994 esittelemä palvelinpuolen ohjelmointikieli, joka on suunniteltu eritoten dynaamisten web-palveluiden tuottamiseen. PHP:n avulla saadaan luotua HTML-sisältöä ja toiminnallisuutta (Tatroe, MacIntyre & Lerdorf 2013, 1), sekä se mahdollistaa tietokantojen tehokkaan käytön palvelimen ja käyttäjän välillä.

Ennen kuin PHP:lla voidaan luoda dynaamista sisältöä, vaatii se toimiakseen web-palvelimen, joka sisältää tuen PHP:lle. Nykyään valtaosa palve-

limista sisältää tuen PHP:lle (The PHP Group 2013), joten käyttäjän näkökulmasta sen käyttäminen on vaivatonta.

PHP:lla ohjelmoitaessa toiminnallisuuden toteuttava koodi sijoitetaan <?php-tagin sisälle ja tiedosto nimetään php-päätteiseksi. Koska PHP-koodi suoritetaan jo palvelimella, ei selaimen tukea PHP:lle vaadita. (Laaksonen 2011.) PHP-sivuilla käytetään HTML-koodia tiedon esittämisessä, sillä käyttäjä ei voi nähdä suoritettavaa koodia selaimellaan.

Kuviossa 2 on havainnollistettu dynaamisen PHP-sivun käsittelyprosessi. Käyttäjän selaimen lähettäessä pyynnön palvelimelle, palvelin lähettää pyydetyn PHP-sivun käännettäväksi palvelimella sijaitsevalle PHP-tulkille, joka prosessoi PHP-koodin ja palauttaa sivun HTML-sivuna takaisin palvelimelle. Tämän jälkeen palvelin palauttaa HTML-sivun käyttäjän selaimelle, jolloin käyttäjä voi lukea PHP-tulkin suorittaman koodin HTML-muodossa. (Murach & Harris 2010, 8.)



Kuvio 2. Dynaamisen PHP-sivun käsittelyprosessi

Esimerkissä 4 on esitetty PHP-koodi, joka tulostaa käyttäjälle lauseen ”Hämeen ammattikorkeakoulu”. Jotta HTTP-palvelimella sijaitseva PHP-tulkki voisi lukea sivun, täytyy tiedostomuotona käyttää php-päätettä.

```
<!DOCTYPE html>
<html>
 <head>
  <title>PHP Esimerkki</title>
 </head>
 <body>
  <?php echo '<p>Hämeen ammattikorkeakoulu</p>'; ?>
 </body>
</html>
```

Esimerkki 4. PHP:n echo-komento, joka tulostaa käyttäjälle halutun lauseen

2.4.1 PDO

PDO eli PHP Data Objects, on PHP-kirjasto, joka luo oikein käytettynä turvallisen ja helppokäyttöisen tavan yhdistää PHP-sovellus tietokantaan. Abstraktitason tietokantametodien ansiosta PDO sopii käytettäväksi melkein kaikkien tietokantaohjelmistojen kanssa. PDO tarjoaa helppokäyttöi-

set ja selkeät luokat ja metodit, joilla tietokantaa käsitellään (Doyle 2010, 359).

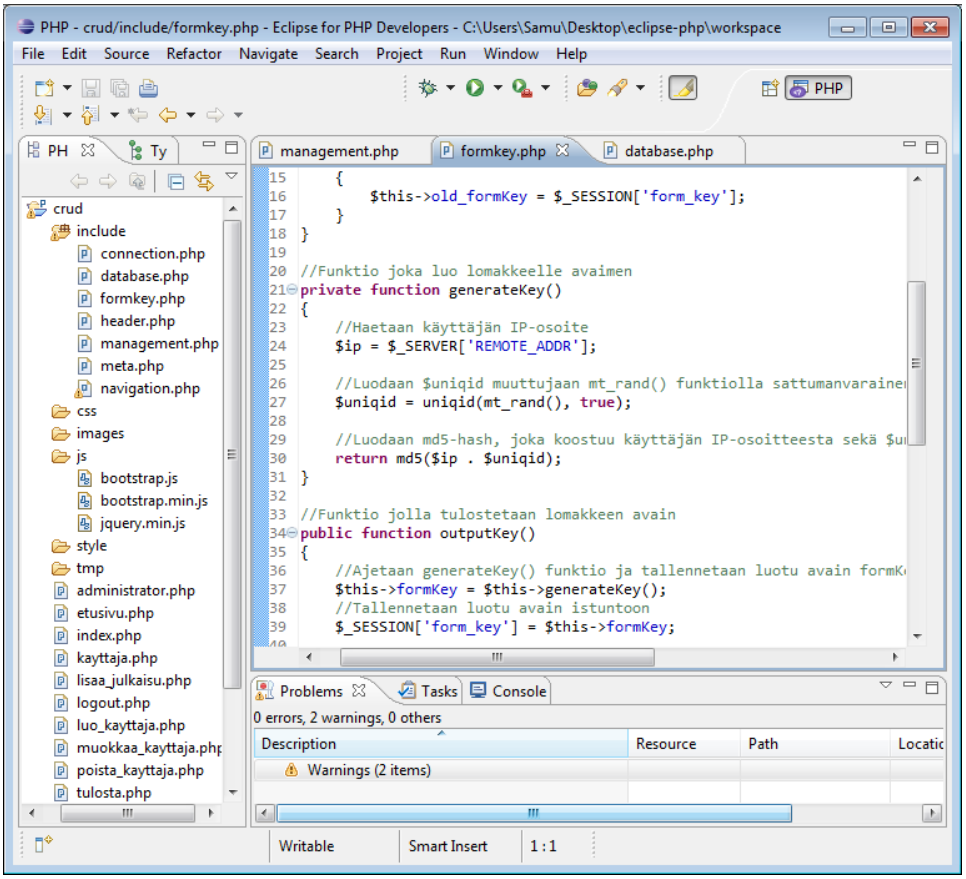
PDO mahdollistaa esimerkiksi muuttujien sitomisen SQL-lausekkeisiin. Muuttujien sitominen suoraan SQL-lausekkeisiin estää käytännössä SQL-injektion mahdollisuuden, josta kerrotaan lisää tämän työn luvussa 4.1. PDO mahdollistaa myös hakulausekkeiden alustamisen ja monen hakulausekkeen suorittamisen yhdellä metodilla. Taulukossa 1 on esitetty yleisimmin käytetyt PDO:n tarjoamat metodit.

Taulukko 1. Yleisimmät PDO-metodit

Metodi	Toiminto
bindParam	Sitoo muuttujan arvon SQL-lausekkeeseen
execute	Suorittaa SQL-lausekkeen mihin sidottu muuttujia
fetch	Noutaa yhden rivin suoritetusta SQL-lausekkeesta
prepare	Valmistelee SQL-lausekkeen execute-metodille
query	Suorittaa yhden SQL-lausekkeen
rowCount	Palauttaa SQL-lausekkeen suorittaman rivimäärän

2.4.2 PHP Eclipse

PHP Eclipse on ohjelmointiympäristö, joka mahdollistaa web-sovellusten tehokkaan suunnittelemisen ja toteuttamisen PHP-kielellä. Ohjelmointiympäristön käyttö helpottaa ohjelmoimista isoissa projekteissa selkeän käyttöliittymän ja työkalujen ansiosta. PHP Eclipse mahdollistaa myös navigoimisen suoraan ohjelmakoodissa, mikä tekee esimerkiksi virheiden etsimisen vaivattomaksi. Kuvassa 5 on esitetty ruutukaappaus PHP Eclipsen käyttöliittymästä.



Kuva 5. PHP Eclipse-ohjelmointiympäristön käyttöliittymä

2.4.3 Twitter Bootstrap

Twitter Bootstrap on yhteisöpalvelu Twitterin luoma sovelluskehys, joka sisältää kokoelman hyödyllisiä työkaluja ja komponentteja verkkosivuston ulkoasun toteuttamiseen. Sovelluskehys sisältää valmiiksi määriteltyjä CSS-tyyliohjeita ja käyttää merkintäkielenään HTML-kieltä. Valmiiksi määriteltyjen tyyliohjeiden avulla etenkin taulukoiden, painikkeiden, elementtien ja lomakkeiden muotoileminen on helppoa. Sovelluskehys sisältää myös jQuery-kirjaston, jolla sivustolle saadaan lisättyä dynaamista toiminnallisuutta. (Bulten 2009.) Esimerkissä 5 on esitetty erilaisia painikkeiden käyttämiä class-attribuutteja, joiden avulla voidaan määrittää CSS-tyyliohjeet painikkeille. Kuvassa 6 on havainnollistettu esimerkissä 5 esitetty koodi suoritettuna selaimessa.

```
<a class="btn" href="nayta.php">Näytä</a>
<a class="btn btn-success" href="muokkaa.php">Muokkaa</a>
<a class="btn btn-danger" href="poista.php">Poista</a>
```

Esimerkki 5. Painikkeiden käyttämät class-attribuutit

Kuva 6. Esimerkissä 3 esitetty koodi suoritettuna selaimessa

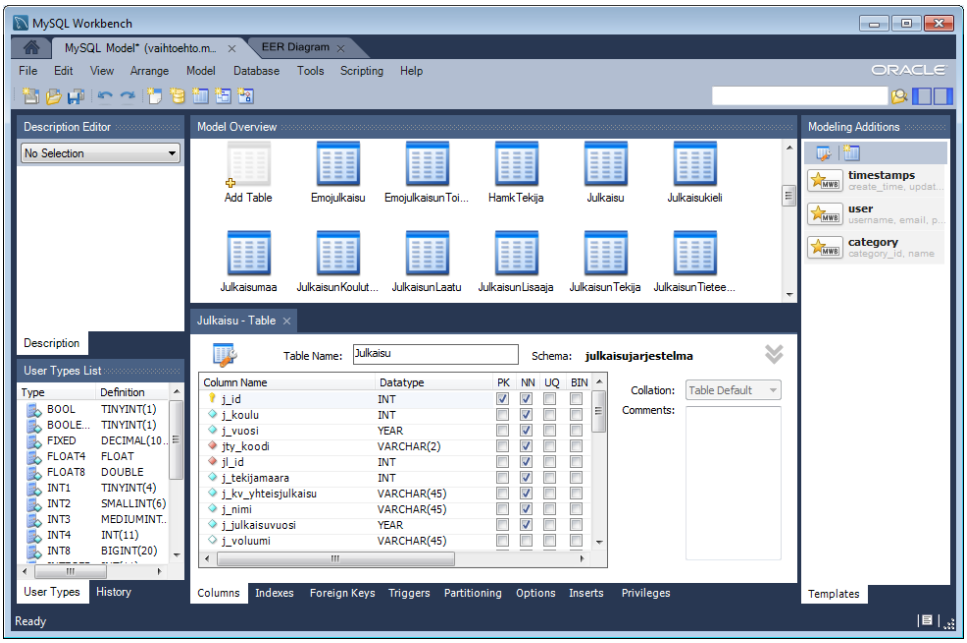
2.5 MySQL

MySQL on Oraclen tarjoama avoimeen lähdekoodiin perustuva relaatiotietokantaohjelmisto, jota käytetään usein web-sovellusten tietokantana. MySQL on hyvin suosittu tietokantaohjelmisto, sillä se on ilmainen ja helppo käyttää. MySQL:stä on olemassa myös maksullinen Enterprise-versio yrityksille. MySQL toimii useimmilla nykyaikaisilla käyttöjärjestelmillä, toisin kuin esimerkiksi Microsoft SQL Server, mikä toimii vain Windows-käyttöjärjestelmällä. (Murach & Harris 2010, 106.)

Siitä huolimatta, että MySQL on ilmainen suurimmalle osalle sen käyttäjistä, tarjoaa se kattavimman tarjonnan ominaisuuksia, joita nykyaikaiselta relaatiotietokantaohjelmistolta voidaan olettaa. Tärkeimpänä näistä on tuki SQL-kielelle sekä usealle yhtäaikaiselle käyttäjälle. (Murach & Harris 2010, 106.)

2.5.1 MySQL Workbench

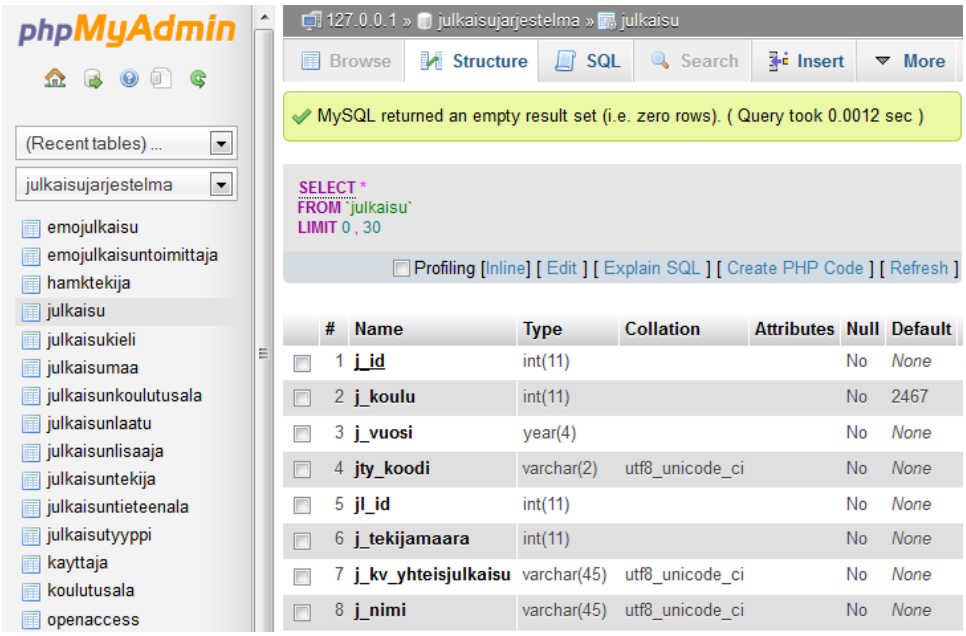
Toimivan ja tehokkaan tietokannan suunnittelu on tarkka ja aikaa vievä prosessi. Tietokantaa suunniteltaessa onkin hyvä käyttää apuna työkalua, joka helpottaa prosessia. Oraclen tarjoama MySQL Workbench on ohjelmisto, joka helpottaa tietokannan suunnittelua ja ylläpitämistä. Se automatisoi virhealttiit ja aikaa vievät työtehtävät ja helpottaa tietokannan visuaalista suunnittelua. Kun tietokanta on suunniteltu, ohjelma luo käyttäjälle tietokannan luontikomennot sekä mahdollistaa suunnitellun tietokannan tallentamisen SQL-muodossa. Kuvassa 7 on esitetty ruutukaappaus MySQL Workbenchin käyttöliittymästä.



Kuva 7. MySQL Workbench-ohjelmiston käyttöliittymä

2.5.2 phpMyAdmin

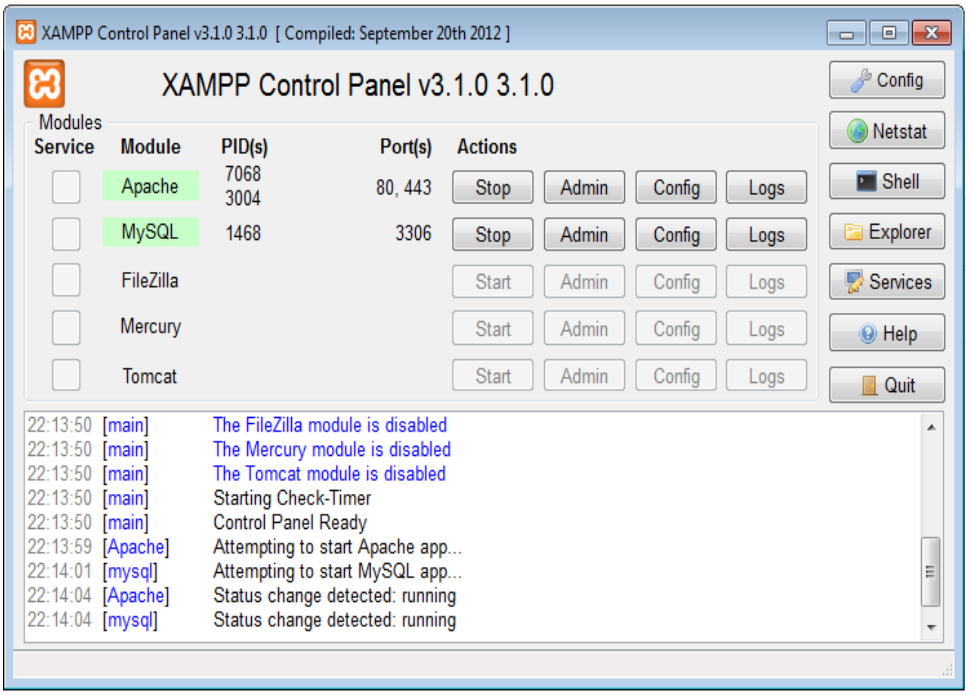
phpMyAdmin on MySQL-tietokannan hallintaan tarkoitettu työkalu, jonka avulla tietokantaa voidaan hallita web-selaimen avulla. phpMyAdmin mahdollistaa tietokantojen luomisen, muokkaamisen ja poistamisen sekä erilaisten komentojen suorittamisen. Kuvassa 8 on esitetty ruutukaappaus phpMyAdminin käyttöliittymästä.



Kuva 8. phpMyAdmin-ohjelmiston käyttöliittymä

2.6 XAMP

XAMP on avoimeen lähdekoodiin perustuva ohjelmisto, jota käytetään yleensä sovellusten testaamisen ja kehittämisen apuvälineenä paikallisella tietokoneella. Ohjelmisto sisältää Apache palvelinohjelmiston ja MySQL-tietokantaohjelmiston sekä käyttää PHP- ja Perl-ohjelmointikielillä kirjoitettuja komentosarjoja. Ohjelma mahdollistaa sovellusten suorittamisen ilman Internet-yhteyttä. Kuvassa 9 on esitetty XAMP-ohjelmiston hallintapaneeli, josta XAMP:n eri moduuleja voidaan käynnistää ja hallinnoida.



Kuva 9. XAMPP-ohjelmiston hallintapaneeli

3 TIETOKANTA

Tietokanta tarkoittaa tietotekniikan sanastossa tietovarastoa, johon voidaan tallentaa tietoa. Tietokannat koostuvat taulukoista, joissa tieto on tallennettu sarakkeisiin. Sarakkeille annetaan tallennettavaa tietoa kuvaava tietotyyppi, jonka avulla tieto saadaan tallennettua oikeassa muodossa ja näin ollen tiedon järjestäminen ja käyttäminen helpottuu. Sarakkeille voidaan myös määritellä tiedon pakollisuus eli voidaanko tietoja tallennettaessa jättää sarake tyhjäksi. Tässä luvussa käsitellään relaatiotietokantoja sekä relaatiotietokantoihin liittyviä käsitteitä.

3.1 Relaatiotietokanta

Relaatiotietokannalla tarkoitetaan tietokantaa, joka koostuu taulujen välisistä relaatioista eli suhteista. Jokaisella taululla on yksilöllinen perusavain, joka voi olla esimerkiksi automaattisesti kasvava numeraalinen arvo. Perusavain ei voi koskaan olla tyhjä, sillä se yksilöi jokaisen taulussa sijaitsevan rivin. Perusavaimeksi tulisi valita arvo, jota ei tarvitse muuttaa sen lisäämisen jälkeen. Perusavain merkitään relaatiokaavioihin yleensä PK-tunnuksella.

Suurin osa relaatiotietokannoista käyttää kyselykieleen SQL-kieltä. SQL-kieli on standardisoitu kyselykieli, jonka ensimmäinen versio kehitettiin IBM:n toimesta 1970-luvulla, relaatiotietokantamallin luojan E.F. Coddin teoriaan pohjautuen. SQL-kielellä tietokantaohjelmistoon lähetetään komentoja, joiden avulla kerrotaan, mikä toiminto halutaan suorittaa. (Huo-

tari 2012, 11–13.) Taulukossa 2 on esitetty yleisimmät SQL-komennot, joita tietokantoihin ja tietokantatauluihin käytetään.

Taulukko 2. Yleisimmät SQL-komennot

Komento	Toiminto
CREATE	Luo uuden tietokannan tai tietokantataulun
SELECT	Valitsee tietoa tietokantataulusta
DELETE	Poistaa tietoa tietokantataulusta
INSERT	Lisää tietoa tietokantatauluun
UPDATE	Päivittää tietoa tietokantataulussa
DROP	Poistaa tietokannan tai tietokantataulun
ALTER	Muokkaa sarakkeita tietokantataulussa

3.2 Taulujen väliset yhteydet

Tietokantataulujen välille voidaan luoda yhteyksiä viiteavaimien avulla. Viiteavaimella tarkoitetaan taulussa olevaa kenttää, joka viittaa toisessa taulussa sijaitsevaan perusavaimeen (Ekonoja, Lahtonen & Mäntylä 2004). Viiteavain merkitään relaatiokaavioihin yleensä FK-tunnuksella. Tietokantataulujen välille voi syntyä kolme erilaista suhdetta eli kardinaalisuutta.

Yhden suhde yhteen -yhteys syntyy, kun yksittäiset tiedot viittaavat taulujen välillä vain yhteen toisessa taulussa olevaan tietoon. Esimerkiksi henkilöllä voi olla vain yksi henkilötunnus, ja yhden henkilötunnuksen voi omistaa vain yksi henkilö.

Yhden suhde moneen -yhteys syntyy, kun toisen taulun tieto voi viitata moneen toisessa taulussa olevaan tietoon. Esimerkiksi henkilöllä voi olla monta luottokorttia, mutta luottokortin omistaa vain yksi henkilö. Tätä yhteyttä käytetään usein tietokantoja normalisoidessa, sillä se poistaa tietoja toistavat sarakkeet tietokantataulusta.

Monen suhde moneen -yhteys syntyy, kun tiedot viittaavat taulujen välillä yhteen tai useampaan toisen taulun tietoon. Esimerkiksi työntekijällä voi olla monta projektia ja projektissa monta työntekijää.

3.3 Viite-eheys

Viite-eheydellä tarkoitetaan yhteyttä, jossa jokainen viittauksen tekevässä taulussa oleva viiteavaimen arvo pysyy samana myös viitattavassa taulussa. Viite-eheydellä varmistetaan, että käytettävä tieto on ensin kirjoitettu viitattavaan tauluun eli isäntä-tauluun, ennen kuin sitä voidaan käyttää viittauksen tekevässä taulussa eli lapsi-taulussa. (Ekonoja, Lahtonen & Mäntylä 2004.)

Otetaan esimerkkinä Kaupunki-taulu, josta viitataan Valtio-tauluun. Tietokannan käyttäjä yrittää luoda kaupunkia, jolle ei ole määritetty valtiota. Viite-eheys huolehtii, että Valtio-tauluun on ensiksi luotava valtio, johon Kaupunki-taulu viittaa, ennen kuin kaupungin luominen voidaan hyväksyä. Viite-eheys ei huolehdi arvoista, jotka voivat olla tyhjiä eli NULL. Tästä syystä ei ole suositeltavaa, että NULL-arvoja käytetään viittauksen tekevässä kentässä. (Ekonoja, Lahtonen & Mäntylä 2004.)

Mikäli on vaarana, että viitattavan taulun viiteavaimena toimivan kentän arvon muuttaminen tai poistaminen aiheuttaa viite-eheyden rikkoutumisen, on huolehdittava, että muutokset välittyvät myös viittauksen tekevään tauluun. Tämä muutos voidaan toteuttaa taulukossa 3 esitetyillä säännöillä.

Taulukko 3. Viite-eheydestä huolehtivia sääntöjä

Sääntö	Toiminto
CASCADE	Perusavaimeen tehtävät muutokset välittyvät viitauksen tekevään viiteavaimeen
RESTRICT	Perusavaimeen ei voi tehdä muutoksia, mikäli siihen on tehty viittauksia
SET DEFAULT	Viittauksen tekevä viiteavain asetetaan oletusarvoonsa, mikäli perusavaimeen tehdään muutoksia
SET NULL	Viittauksen tekevä viiteavain nollataan, mikäli perusavaimeen tehdään muutoksia

3.4 Normalisointi

Tietokannan normalisoinnilla tarkoitetaan prosessia, jossa tietokannan rakenne pilkotaan normaalimuotoja noudattamalla tehokkaasti saatavilla olevaan muotoon. Normalisoinnin tarkoituksena on poistaa tietokannassa helposti tapahtuvaa redundanssia eli saman tiedon toistumista sekä tukea tiedon ehjää tallentamista (Nixon 2012, 207). Tämä saadaan toteutumaan jakamalla tieto pienempiin osiin luomalla uusia tauluja sekä antamalla tauluille pääavaimet. Redundanssi tekee tietokannasta normaalia suurempikokoisen ja näin ollen hidastaa tiedon hakemista.

Täysin normalisoidussa tietokannassa on ehkäisty tiedon toistaminen tehokkaasti ja jokainen tieto on tallennettu vain yhteen paikkaan. Tietokantasovelluksia ohjelmoitaessa tiedon toistamisen ehkäisystä on suuri hyöty,

sillä kirjoitusfunktiot on helppo ohjelmoida oikein, kun samaa tietoa ei kirjoiteta moneen eri paikkaan ja tietokannan rakenne on selkeä. (Downs 2008.)

Relaatiotietokantamallin kehittäjä E.F. Codd analysoi normalisointia ja esitti kolmea normaalimuotoa, joita käyttämällä tietokannan rakenne saatiin optimoitua tukemaan tehokasta käytettävyyttä sekä pientä muistin ja kovalevyn käyttöä (Nixon 2012, 207).

3.4.1 Ensimmäinen normaalimuoto

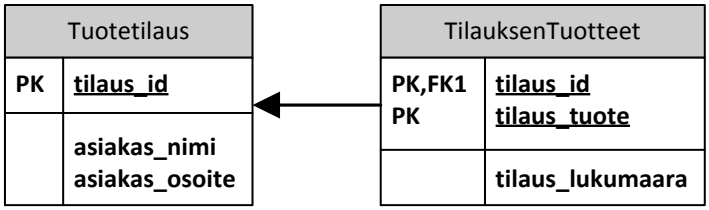
Ensimmäisen normaalimuodon tarkoituksena on poistaa samaa tietoa toistavat sarakkeet sekä poistaa moniarvoiset attribuutit. Normalisointi toteutetaan luomalla moniarvoisille attribuuteille omat taulut. Jokaisella taululla täytyy olla yksilöllinen perusavain, jotta ensimmäisen normaalimuodon ehdot täyttyvät.

Kuvassa 10 on esitetty tietokantataulu, joka rikkoo kaikkia normaalimuotoja. Taulu sisältää samaa tietoa toistavia sarakkeita, kuten tuotteet ja lukumäärät. Mikäli asiakas haluaisi tilata kolme erilaista tuotetta, ei tietokanta pystyisi siihen.

Tuotetilaus	
PK	**tilaus_id**
	asiakas_nimi **asiakas_osoite** **tilaus_tuote1** **tilaus_lukumaara1** tilaus_tuote2 tilaus_lukumaara2

Kuva 10. Tietokantataulu, joka rikkoo kaikkia normaalimuotoja

Ratkaisu ongelmaan on luoda tilaukseen tuleville tuotteille oma taulunsa. Kuvassa 11 on esitetty tietokantaratkaisu, joka noudattaa ensimmäistä normaalimuotoa. Yksilölliset perusavaimet on merkitty tietokantatauluihin PK-lyhenteellä

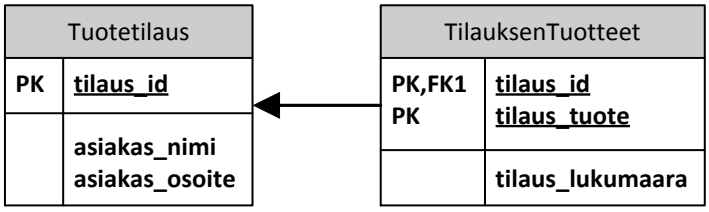


Kuva 11. Ensimmäistä normaalimuotoa noudattava tietokantaratkaisu

3.4.2 Toinen normaalimuoto

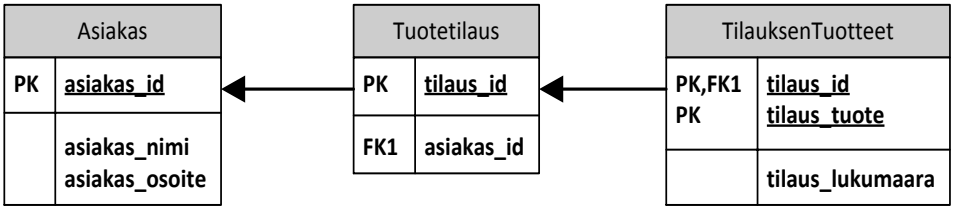
Kun ensimmäinen normaalimuoto poisti redundanssin sarakkeiden välillä, poistaa toinen normaalimuoto sen sijaan rivien välisen redundanssin. Redundanssi saadaan poistettua etsimällä toistuvia tietoja tietokantataulusta ja siirtämällä ne omiin tauluihinsa. Toinen normaalimuoto vaatii myös, että kaikki sarakkeet, jotka eivät ole avaimia, ovat funktionaalisesti riippuvaisia taulun avaimesta. Funktionaalisella riippuvuudella tarkoitetaan yhteyttä, jossa tietämällä arvo A tiedetään myös arvo B. (Saarelainen 2008b, 3–4.)

Kuvasta 12 voidaan havaita tiedon toistumista riveillä. Myöskään osoite ei ole funktionaalisesti riippuvainen tilaus_id-perusavaimesta, vaan osoite on yhteydessä tilaajan nimeen. Tilaajan nimi ei voi olla perusavain, koska nimi ei ole yksilöllinen, toisin kuin esimerkiksi henkilötunnus.



Kuva 12. Toista normaalimuotoa rikkova tietokantaratkaisu

Ratkaisu on luoda asiakkaille oma taulu, joka yhdistetään tuotetilaustaulusta asiakastauluun viiteavaimen avulla. Viiteavaimeksi valitaan asiakastaulun id, joka on yksilöllinen, toisin kuin esimerkiksi asiakkaan nimi tai osoite. Kuvasta 13 voidaan havaita kuvassa 12 esitetyn tietokantataulun pohjalta luotu, toista normaalimuotoa noudattava tietokantaratkaisu.



Kuva 13. Ensimmäistä ja toista normaalimuotoa noudattava tietokantaratkaisu

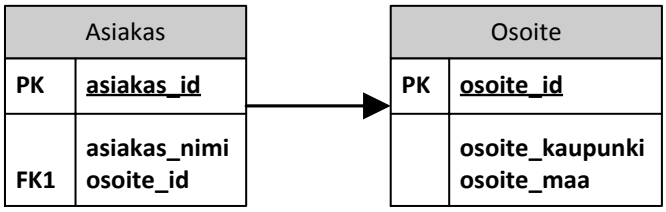
3.4.3 Kolmas normaalimuoto

Kolmannen normaalimuodon mukaisessa tietokannassa kaikkien sarakkeiden tulee riippua pääavaimesta. Sen lisäksi ensimmäinen ja toinen normaalimuoto täytyy olla saavutettu. Kolmas normaalimuoto saavutetaan siirtämällä kaikki pääavaimesta riippumattomat sarakkeet omiin tauluihinsa.

Kuvassa 14 on esitetty tietokantataulu, johon on tallennettu asiakkaan nimi, kaupunki sekä valtio, missä kaupunki sijaitsee. Koska maa ei ole riip-

puvainen asiakkaan nimestä, vaan kaupungista, jossa asiakas sijaitsee, siirretään kaupunki ja maa omaan tauluunsa kuvan 15 osoittamalla tavalla.

Kuva 14. Kolmatta normaalimuotoa rikkova tietokantataulu



Kuva 15. Kaikkia kolmea normaalimuotoa noudattava tietokantaratkaisu

3.5 Denormalisointi

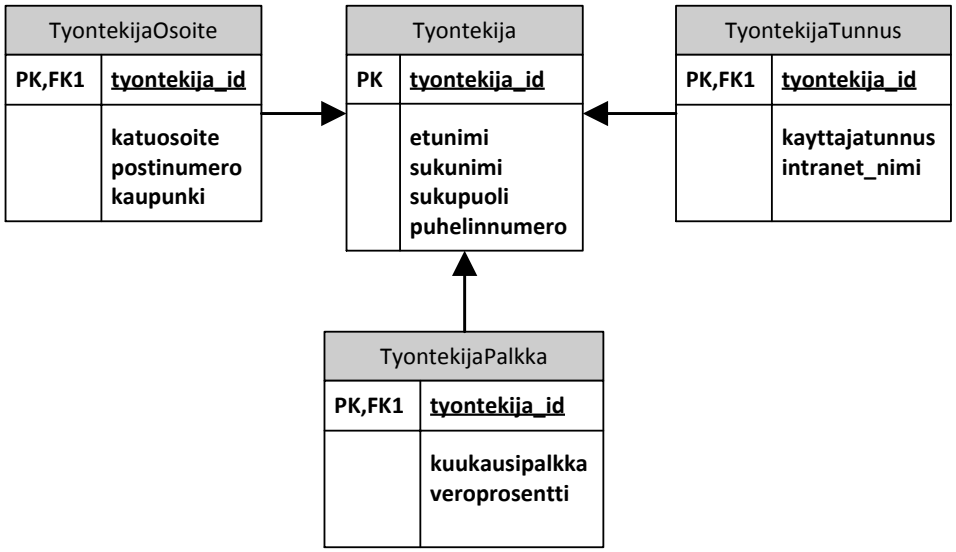
Denormalisoinnilla tarkoitetaan prosessia, jossa tietokannan suorituskykyä optimoidaan yhdistämällä toisiinsa liittymättömiä tietoja samaan tietokantatauluun tai yhdistämällä tietokantatauluja toisiinsa. Denormalisointi rikkoo normalisoinnissa esitettyjä normaalimuotoja, sillä denormalisoidessa tapahtuu tiedon toistamista eli redundanssia. Kun puhutaan suorituskyvystä lukea tietokantoja, on syytä mainita, että nykyaikainen tietokone pystyy lukemaan iso- tai pienikokoisen, normalisoidun tai denormalisoidun tietokannan lähes samalla nopeudella. (Atwood 2008.)

Denormalisointia kannattaa käyttää eritoten tilanteissa, jolloin täysin normalisoitu tietokanta aiheuttaa haittaa tietokannan tai sovelluksen toiminnalle. Esimerkkeinä tilanteista, jolloin tietokannan denormalisointia kannattaa harkita, voidaan mainita laskutoimitukset sekä monimutkaiset JOIN-lausekkeet. (Downs 2008.)

Laskutoimituksia tehdessä voi tuntua paremmalta idealta luoda laskutoimitukselle oma tietokenttä. Otetaan esimerkkinä kauppa, joka tulostaa laskun asiakkaalle. Laskuun tulostettavat summat riippuvat muun muassa tuotteiden määrästä, hinnasta, verosta ja muista laskutukseen liittyvistä tekijöistä. Oletetaan, että tietokanta on täysin normalisoitu ja laskutoimitukset tehdään ohjelmallisesti ja laskutoimitukset voivat olla hyvinkin monimutkaisia ja summa voi koostua monesta eri laskutoimituksesta. Mikäli ohjelmaan tehdään päivityksiä tai rakenteellisia muutoksia, syntyy helposti tilanteita, jolloin laskuun tulostuu virheellisiä summia, sillä ohjelman rakenne on muuttunut ja laskutoimitukset voivat käyttävää vääriä kenttiä tai arvoja. Tämä ongelma on helppo ratkaista rikkomalla normalisoinnin

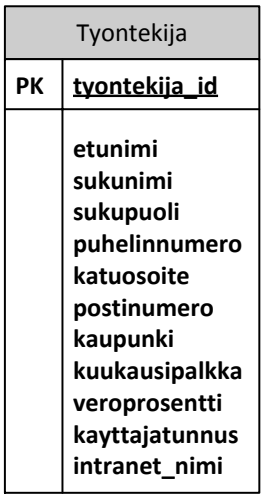
kolmatta normaalimuotoa tekemällä tietokantatauluun laskutoimituksien summille omat tietokentät, jolloin laskuun saadaan tulostettua arvo käyttämällä suoraan yhtä kenttää tietokannasta.

Monimutkaiset JOIN-lausekkeet aiheuttavat myös ongelmia täysin normalisoiduissa tietokannoissa. Koska tietokanta on pilkottu moneen tauluun, on tiedon syöttäminen tietokantaan helppoa, mutta sen sijaan lukeminen voi olla hyvinkin työlästä. Otetaan esimerkkinä kuvassa 16 esitetty tietokanta, johon tallennettaan tiedot yrityksen työntekijöistä. Kolmatta normaalimuotoa noudattavassa tietokannassa työntekijän perustiedot, osoitetiedot, käyttäjätunnuksiin liittyvät tiedot sekä työntekijän palkkaan liittyvät tiedot olisi tallennettu erillisiin tauluihin. Esimerkin tapauksessa käyttäjälle halutaan tulostaa kaikki työntekijään liittyvät tiedot, jolloin tarvittaisiin neljä JOIN-lauseketta haun toteutumiseen. Tämä haku ei paranna tietokannan suorituskykyä ja tämän lisäksi monimutkaiset JOIN-lausekkeet ovat hyvin herkkiä rakennemuutoksille.



Kuva 16. Kolmatta normaalimuotoa noudattava tietokantaratkaisu

Kuvasta 17 voidaan havaita denormaliosidun tietokantaratkaisun kuvassa 16 esitettyyn tietokantaratkaisuun. Koska kaikki työntekijän tiedot ovat denormalisoitu yhteen tietokantatauluun, tarvitaan tiedon hakemiseen vain yksinkertainen SELECT-lauseke, joka hakee kaikki Tyontekija-taulun tiedot.



Kuva 17. Kuvassa 16 esitetty tietokantaratkaisu denormalisoituna

Normalisoinnista poikkeaminen täytyy tehdä aina hallitusti ja perustellusti, sillä denormalisoinnilla on hintansa (Pulkkinen 2012). Tietokannan normalisointi on yleensä suositeltavaa suurimmassa osassa tietokannoista, sillä se parantaa suorituskykyä poistamalla redundanssin ja opettaa oikeaoppiseen tietokantasuunnitteluun, mutta on tärkeää, että tietokannan suunnittelija ymmärtää, milloin tietokanta kannattaa denormalisoida. Nyrkkisääntönä voidaan sanoa, että ensiksi kannattaa normalisoida niin kauan, kun siitä ei ole haittaa suorituskyvylle ja tämän jälkeen denormalisointi kannattaa niin kauan, kun tietokanta toimii ja denormalisoinnista on käytännön hyötyä. (Atwood 2008.)

3.6 Näkymät

Näkymät tietokannoissa voidaan ajatella eräänlaisina tallennettuina kyselyinä. Näkymä tallentaa näkymän luomisessa käytetyn hakulausekkeen virtuaaliseen tauluun, jonka jälkeen taulua voidaan käyttää normaalin taulun tapaan. Näkymän tarkoituksena on helpottaa toistuvien kyselyiden tekemistä ja se toimii myös turvallisuutta parantavana tekijänä, kun tietoa ei tarvitse toistuvasti noutaa monimutkaisilla SQL-lausekkeilla. (Saarelainen 2008a, 1-3.) Esimerkissä 6 on esitetty SQL-lauseke, jolla luodaan kirjat-näkymä. Näkymän haetaan kaikki tiedot julkaisuista, joiden julkaisutyyppinä on kirja.

```
CREATE VIEW kirjat AS SELECT * FROM julkaisut WHERE julkai-
sutyyppi = 'kirja';
```

Esimerkki 6. Näkymän luominen

4 TIETOTURVA

Silloin, kun tieto on helposti saatavilla, on olemassa riski tiedon väärinkäytölle. Mikäli tiedon saatavuudesta ei tarvitsisi huolehtia, ei sen turvallisuuskaan olisi huolenaihe. Tietoturva on käsitteenä hyvin laaja, mutta käy-

tännössä se tarkoittaa tiedon väärinkäytön estämistä. Väärinkäytöksi voidaan laskea tiedon vuotamista henkilöille joilla ei ole siihen oikeutta, tiedon tahallista tuhoamista tai muuntamista sekä tiedon saatavuuden estämistä. (Kuivanen 2005.)

Viimeisin suuri tietomurto Suomessa ilmeni syyskuussa 2013, kun viestintäviraston tietoturvayksikkö CERT-FI julkaisi tiedotteen, jossa kerrottiin yli sadan palvelun käyttäjätietojen vuotamisesta. Hyökkäykset oli toteutettu käyttäen SQL-injektio ja Cross-site scripting haavoittuvuuksia järjestelmissä. (Viestintävirasto 2013.)

PHP:n tietoturvan kannalta kaikista tärkeintä on olla luottamatta käyttäjän syötteeseen (Snyder, Myer & Southwell 2010, 10). Käyttäjän syötteellä tarkoitetaan kaikkea tietoa, mitä käyttäjä pystyy lisäämään järjestelmään. Esimerkiksi SQL-injektiot ja Cross-site scripting-hyökkäykset ovat mahdollisia vain, mikäli käyttäjän syötettä ei tarkisteta tarpeeksi hyvin. Vaikka käyttäjän syötteen tarkistaminen olisi toteutettu tietoturvallisesti, niin on myös tärkeää ottaa huomioon muun tyyppiset hyökkäykset, kuten selainpohjainen Cross-site request forgery-hyökkäys, joka käyttäjän syötteen sijaa käyttää hyväksi käyttäjän selainta haitallisten komentojen syöttämiseen (Zeller 2008).

4.1 SQL-injektio

SQL-injektio on yksi yleisimmistä haavoittuvuuksista PHP:lla ohjelmoiduissa järjestelmissä. SQL-injektiolla tarkoitetaan hyökkäystä, jossa tietokantaan syötetään hyökkääjän toimesta haitallisia SQL-lausekkeita (Snyder, Myer & Southwell 2010, 45). SQL-injektion riski syntyy, kun tiedon syöttämistä ei suodateta ennen kuin se siirtyy järjestelmään ja kun erikoismerkkejä ei poisteta tiedon siirtyessä järjestelmästä tietokantaan. SQL-injektion tekeminen vaatii kuitenkin taitoa ja kokeilemista hyökkääjältä, sillä hyökkääjän on tiedettävä tai arvattava tietokantataulun rakenne, olettaen, ettei hyökkääjällä ole suoraa pääsyä lähdekoodiin tai tietokantakaavioon. (Shiflett 2006, 35.)

Esimerkissä 7 on havainnollistettu SQL-injektio. Oletetaan, että syötettyjä tietoja ei suodateta, vaan ne siirretään tietokantaan siinä muodossa, missä käyttäjä on ne syöttänyt. Käyttäjän lähettämältä lomakkeelta saapuvat muuttujat tunnus ja salasana. SQL-injektion tekevä henkilö syöttää käyttäjätunnukseksi admin ja salasanaksi ' OR '' = ' –ehdon. Syötetyn ehdon mukaan tyhjän arvo täytyy vastata tyhjää arvoa. Näin ollen OR-ehto toteutuu ja SQL-injektio onnistuu. (Laaksonen 2010.)

```
SELECT * FROM users
WHERE tunnus = '$tunnus' AND salasana = '$salasana'

SELECT * FROM users
WHERE tunnus = 'admin' AND salasana = ' OR '' = '
```

Esimerkki 7. SQL-injektio users-tauluun

SQL-injektion mahdollisuus saadaan poistettua käyttämällä muuttujia suoraan SQL-lausekkeissa ja valmistelemalla lausekkeet ennen niiden suorittamista. Esimerkissä 7 esitetyn SQL-injektion estämiseen riittää esimerkissä 8 esitetty ratkaisu jossa käytetään PDO:n tarjoamia prepare-, bindParm- ja execute-metodeja, joilla SQL-lauseke saadaan valmisteltua sekä muuttujat sidottua lausekkeeseen. Lauseketta ei siis suoriteta suoraan, vaan se valmistellaan ensin execute-metodille suoritettavaksi, jolloin lauseke on suojattu SQL-injektiolta.

```
$sql = 'SELECT * FROM users WHERE tunnus = :tunnus AND
salasana = :salasana';
$result = $this->pdo->prepare($sql);
$result->bindParam(':tunnus', $tunnus);
$result->bindParam(':salasana', $salasana);
$result->execute();
```

Esimerkki 8. SQL-injektiolta suojattu kirjautuminen

4.2 Cross-site scripting

Cross-site scripting eli XSS-hyökkäys on yksi tunnetuimmista hyökkäystavoista web-sovelluksia vastaan (Shiflett 2006, 23). Siinä missä SQL-injektiossa yritetään syöttää haitallisia SQL-lausekkeita järjestelmään ilman, että käyttäjät näkevät niitä, niin XSS-hyökkäyksessä haitallinen koodi syötetään käyttäjän nähtäväksi verkkosivulle (Snyder, Myer & Southwell 2010, 45). Nimensä mukaisesti haitallinen koodi syötetään toiselta verkkosivulta sivujenvälisesti toimivana komentosarjana kohteena olevaan web-sovellukseen, jolloin hyökkäys näyttää web-sovelluksen näkökulmasta tulevan luotettavasta lähteestä eli järjestelmän käyttäjältä itseltään (Ballad & Ballad 2009, 137). Hyökkääjälle tämä mahdollistaa käyttäjäpuolen turvallisuuden kiertämisen, esimerkiksi kirjautumisen, mikäli toinen käyttäjä on valmiiksi kirjautunut järjestelmään.

XSS-hyökkäys toteutetaan yleensä käyttäen asiakaspuolen komentosarjakieliä, kuten esimerkiksi JavaScriptia, mutta sen toteuttamiseen voidaan käyttää myös HTML-kieltä (Ballad & Ballad 2009, 137). XSS-hyökkäykset voidaan lajitella pysyviin ja ei-pysyviin hyökkäysiin.

Ei-pysyvällä hyökkäyksellä tarkoitetaan hyökkäystä, joka ei tallenna tietota järjestelmään, vaan toimii sen sijaan käyttäjän selaimessa. Käyttäjä voi esimerkiksi painaa linkkiä, joka yhdistää halutulle sivustolle, mutta linkin sekaan on syötetty haitallista koodia, joka auttaa hyökkäyksen tekijää saamaan haluamansa tiedot. Sivu voi näyttää täysin oikealta, mutta haitallisen koodin ansiosta hyökkääjä voi esimerkiksi syöttää käyttäjälle haitallisia evästeitä, ja käyttää niitä myöhemmin käyttäjän istunnon varastamiseen. (Ballad & Ballad 2009, 137–138.)

Pysyvällä hyökkäyksellä tarkoitetaan hyökkäystä, joka tallentaa tietoa järjestelmään. Onnistuessaan pysyvät hyökkäykset ovat vaarallisempia,

koska sen sijaan, että ne vaikuttaisivat yhteen käyttäjän, ne vaikuttavat kaikkiin järjestelmän käyttäjiin. (Ballad & Ballad 2009, 138.) Keskustelufoorumit, vieraskirjat sekä muut verkkosivustot, joihin käyttäjä pystyy lisämään tietoa muiden käyttäjien näkyville, ovat eritoten alttiita pysyville hyökkäyksille. Hyökkäyksen tekevä käyttäjä pystyy esimerkiksi lisäämään linkin foorumille, joka lähettää linkin avaavan käyttäjän kirjautumistiedot hyökkääjälle.

XSS-hyökkäys estetään poistamalla HTML-merkintäkieli esitettävästä tekstistä. Tämä toteutetaan käyttämällä PHP:n tarjoamaa htmlspecialchars-funktiota. Esimerkissä 9 on esitetty, miten kyseistä funktiota käytetään. Oletetaan, että XSS-hyökkäyksen tekijä on lisännyt sivustolle kommentin, mikä sisältää HTML-merkkejä sisältävän komentosarjan. Komentti asetetaan vanha-muuttujaan ja tallennetaan tietokantaan. Mikäli HTML-merkintöjä ei poistettaisi muuttujasta ennen sen näyttämistä verkkosivulla, voisi kommentti sisältää esimerkiksi linkin, jota painamalla komentosarja lähettää käyttäjän tiedot hyökkääjälle. Kun vanha-muuttuja myöhemmin haetaan tietokannasta, käytetään htmlspecialchars-funktiota HTML-merkkien poistamiseen. Mikäli vanha-muuttuja sisältäisi esimerkiksi tekstin ”<script>”, olisi funktion käyttämisen jälkeen uusi-muuttujaan tallennettava teksti ”<script>”.

```
$vanha = $_POST['kommentti'];
$uusi = htmlspecialchars($vanha);
```

Esimerkki 9. Muuttujan sisällöstä poistetaan erikoismerkit htmlspecialchars-funktiolla

4.3 Cross-site request forgery

Cross-site request forgery eli lyhennettynä CSRF, on hyökkäys, jossa käytetään hyväksi web-sovelluksen luottamusta käyttäjän selaimelta tuleviin käskyihin. CSRF-havoittuvuus syntyy, kun web-sovellukseen lähetetään käyttäjältä käskyjä, esimerkiksi lomakkeiden avulla, mutta alkuperää ei tarkisteta. Yleensä web-sovellukset eivät tarkista, että käsky tuli käyttäjältä itseltään, vaan sen sijaan selaimet varmistavat, että käsky tuli käyttäjän selaimesta. Koska käsky tulee käyttäjän selaimesta, on CSRF-haavoittuvuuksien estäminen tarpeellista myös paikallisessa verkossa toimivissa järjestelmissä. (Zeller 2008.)

Esimerkissä 10 on esitetty esimerkki CSRF-hyökkäyksestä. Oletetaan, että järjestelmään kirjautunut käyttäjä vierailee verkkosivulla, johon on tallennettu kuvalinkki. Kun järjestelmään kirjautunut käyttäjä vierailee sivustolla, selain suorittaa sivun HTML-koodin. Selain ei tunnista, että kuvalinkki ei ole linkki kuvaan vaan verkkosivulle. Selain yrittää lähettää linkissä olevaan osoitteeseen käskyn poistaa käyttäjä, jonka id on 35. Mainittakoon, että CSRF-hyökkäyksen tekeminen ei ole helppoa, sillä hyökkäyksen tekevän täytyy tietää oikeat tiedostonimet ja niissä käytettävät muuttujat.

```
<img src="http://verkkosivu.fi/poista_kayttaja.php?id=35"/>
```

Esimerkki 10. CSRF-hyökkäyksen toteuttava kuvalinkki

CSRF-hyökkäys voidaan estää luomalla generoitu avain jokaiseen lomakkeeseen millä tietoa lähetetään. Näin varmistetaan, että lomakkeelta tulevat tiedot tulevat halutulta sivustolta. Lomakkeiden suojaamista käsitellään tämän työn luvussa 6.2.

4.4 Salasanat

Tietoturvallisista syistä salasanoja ei tulisi ikinä tallentaa tietokantaan ilman salausta, sillä jos järjestelmä ei ole suojattu tarpeeksi hyvin, voi taitava hakkeri saada haltuunsa tietokantataulun sisältöä SQL-injektion tai XSS-hyökäyksen avulla. Tietoturvallinen ratkaisu salasanojen tallentamiseen tietokantaan on salata tiedot ennen niitten syöttämistä. Salaamiseen voidaan käyttää niin kutsutta suolaus-tekniikkaa eli salasanaan lisätään koodillisesti annettu merkkijono.

Esimerkissä 11 voidaan havaita miten suolaus-tekniikkaa käytetään. Ensimmäiseksi käyttäjän syöttämä salasana haetaan POST-metodista ja tallennetaan saatu arvo salasana-muuttujaan. Tämän jälkeen password-muuttujalle annetaan arvoksi suolauksessa käytettävä merkkijono ja luodaan merkkijonosta MD5-tiiviste. Luotu MD5-tiiviste salataan SHA-1-salaustekniikalla ja tallennetaan salt-muuttujaan. Viimeiseksi salasana- ja salt-muuttujiin tallennetut arvot yhdistetään ja niistä luodaan MD5-tiiviste ja saatu tiiviste tallennetaan md5-muuttujaan. Tämä arvo myöhemmin tallennetaan myös tietokantaan.

```
$salasana = $_POST['salasana'];
$password = "suolaus";
$salt = sha1(md5($password));
$md5 = md5($salasana.$salt);
```

Esimerkki 11. Salasana suojaaminen käyttäen suolaus-tekniikkaa sekä SHA-1- ja MD5-salaustekniikoilla

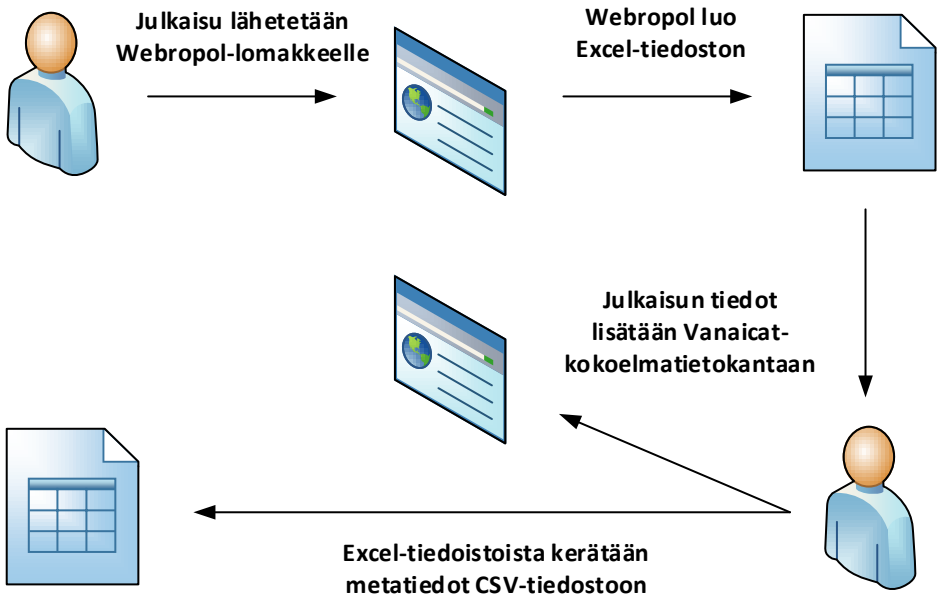
Esimerkin 11 tapauksessa tietokantaan tallentuisi salasanaksi merkkijono ”bb83b23b415a340f989b420a0351ba97”, mikäli salasanana olisi ”salasana” ja suolauksena sana ”suolaus”. Tietokantataulun joutuessa vääriin käsiin, alkuperäisiä salasanoja ei saataisi selville ennen kuin salauksessa käytetyt salaustekniikat ja suolauksessa käytetty merkkijono olisi selvillä.

5 SUUNNITTELU

Vuodesta 2012 lähtien opetus- ja kulttuuriministeriö on vaatinut ammattikorkeakouluja lähettämään metatiedot ammattikorkeakouluissa tehdyistä julkaisuista (Liite 1). Opetus- ja kulttuuriministeriön vaatimat metatiedot ovatkin aiheuttaneet ongelmia ammattikorkeakouluille, sillä monessa ammattikorkeakoulussa julkaisurekisterit on toteutettu osana Vanaicat-kokoelmatietokantaa. Ongelmana on, että kokoelmatietokantaan ei voida

tallentaa kaikkia vaadittuja metatietoja ilman, että kokoelmatietokannan ensisijainen käyttötarkoitus häiriytyisi oleellisesti.

Työn toimeksiantajalla on käytetty vuodesta 2012 lähtien tiedonkäsittelyprosessia, jossa toimeksiantajalla tehtyjen julkaisujen tiedot tallennetaan Webropol-verkkolomakkeella Excel-tiedostoon, josta julkaisurekisteriin vaadittavat metatiedot siirretään käsin Vanaicat-kokoelmatietokantaan. Webropol-verkkolomakkeella kerätään myös opetus- ja kulttuuriministeriön vaatimat metatiedot, mutta näitä metatietoja ei tallenneta kokoelmatietokantaan, vaan tallentamisen sijaan metatiedot lähetetään vuosittain kahtena CSV-tiedostona opetus- ja kulttuuriministeriölle. Metatiedot sisältävien CSV-tiedostojen luominen on kuitenkin virhealtis ja ajallisesti hidas prosessi, sillä metatiedot kerätään käsin Excel-tiedostosta. Kuviossa 3 on havainnollistettu tällä hetkellä käytössä oleva tiedonkäsittelyprosessi.



Kuvio 3. Vuodesta 2012 lähtien käytetty tiedonkäsittelyprosessi

Käytännön osuuden tarkoituksena on tuottaa toimeksiantajalle julkaisujärjestelmä, joka tulisi korvaamaan tällä hetkellä käytössä olevan tiedonkäsittelyprosessin. Tiedonkäsittelyprosessi, johon työssä toteutettavalla julkaisujärjestelmällä pyritään pääsemään, on havainnollistettu kuviossa 4. Kuvassa esitetyssä tiedonkäsittelyprosessissa käyttäjä lisää julkaisun suoraan työssä toteuttavaan järjestelmään, josta syötettyjä tietoja voidaan lukea, muokata ja poistaa. Järjestelmästä pystytään tallentamaan julkaisujen tiedot suoraan CSV-tiedostoon käyttäjän määrittelemän julkaisutyyppiluokituksen ja vuoden mukaan. Näin toimimalla toimeksiantaja säästää aikaa ja työtunteja, sekä vältytään tiedonsiirrossa mahdollisesti tapahtuvilta inhimillisiltä virheiltä.

Kuvio 4. Tiedonkäsittelyprosessi johon julkaisujärjestelmällä pyritään pääsemään

Järjestelmän suunnittelun tietoperustana käytettiin työn toteuttajan aiemmin hankkimaa kokemusta ja tietoa järjestelmien suunnittelusta, sekä toimeksiantajan luomaa Wikiä, joka toimi myös kommunikointivälineenä työn toteuttajan ja toimeksiantajan välillä. Järjestelmän suunnitteluvaiheessa pyrittiin kartoittamaan järjestelmän toiminnalliset ja laadulliset vaatimukset, sekä luotiin käyttöliittymäluonnokset. Kerätyn tiedon pohjalta suunniteltiin tietokanta, joka täytti järjestelmän vaatimukset ja mahdollisti järjestelmän optimaalisen toiminnan.

5.1 Toiminnalliset vaatimukset

Järjestelmän toiminnalliset vaatimukset määriteltiin yhdessä toimeksiantajan kanssa. Toimeksiantajan kanssa käydyissä keskusteluissa selvitettiin järjestelmältä vaadittuja ja toivottuja toiminnallisia ominaisuuksia sekä luotiin selkeä näkemys päämäärästä toimeksiantajan ja työn toteuttajan välille.

Järjestelmän tärkeimpiä vaadittuja ominaisuuksia oli mahdollisuus tallentaa julkaisut vuosittain CSV-tiedostoon. Käyttäjän pitää pystyä itse määrittelemään vuosiluku ja julkaisutyyppiluokka tiedostoja tallentaessa. CSV-tiedostoon tallennettavien julkaisujen metatiedot täytyy olla koodattu UTF 8-merkistökoodauksella, jotta opetus- ja kulttuuriministeriön järjestelmä pystyy vastaanottamaan tiedot. Tiedot tulisi myös tallentaa niin, että työntekijän täytyy tehdä mahdollisimman vähän käsin tehtävää tiedonmuokkausta järjestelmässä.

Toinen tärkeä vaatimus järjestelmältä oli, että julkaisun tekijä voi ilmoittaa julkaisunsa tiedot järjestelmään, jonka jälkeen ilmoittaja ja kirjaston vastuuhenkilö saavat sähköpostitse ilmoituksen julkaisun lisäämisestä järjestelmään. Kirjaston vastuuhenkilö tarkistaa syötetyt tiedot, ja tekee mahdolliset korjaukset tietoihin. On tärkeää, että julkaisun lisääjän tiedot lisätään järjestelmää, jotta yhteydenotto julkaisun lisääjään on tarvittaessa mahdollista.

Järjestelmältä toivottuja ominaisuuksia oli tekijöiden ja kokoelmateosten selauksen mahdollistaminen. Lisätty julkaisu voi olla osa kokoelmateosta joten kokoelmateoksia selaamalla näkisi kuhunkin kokoelmateokseen sisältyvät julkaisut. Julkaisulla voi olla myös monta tekijää ja tekijöillä

monta julkaisua, joten mahdollisuus tekijöiden kaikkien julkaisujen selaamiseen olisi hyvä olla olemassa.

Muita toivottuja ominaisuuksia oli mahdollisuus raportoida järjestelmän julkaisuista löytyviä virheitä, sekä julkaisuja lisätessä opetus- ja kulttuuriministeriön laatimat luokitusten mukaisesti annettavat tiedot olisi valittavissa pudotusvalikoista. Näitä tietoja ovat esimerkiksi koulutus- ja tieteenalat sekä julkaisumaat, joille on määrätty ennalta luokkakoodit ja nimet.

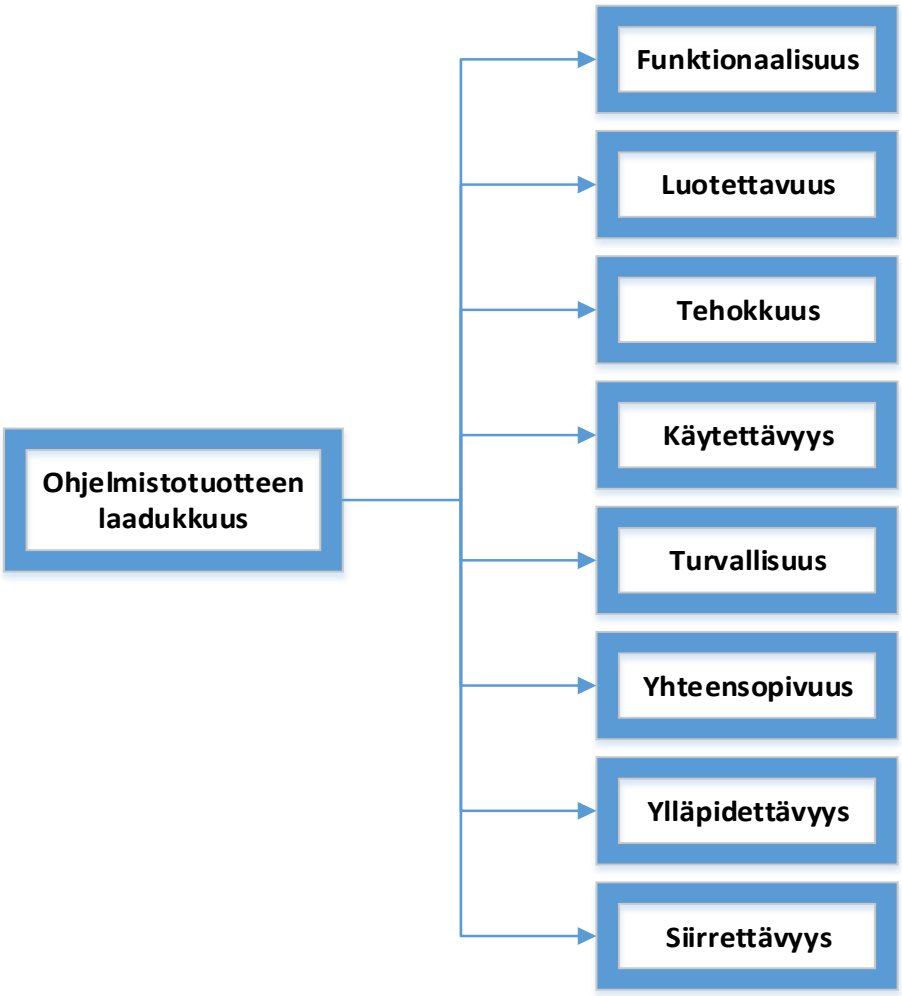
5.2 Laadulliset vaatimukset

Järjestelmän laadullisia vaatimuksia määritellessä käytettiin apuna ISO/IEC 25010-standardin mukaisia ohjelmistotuotteen laadukkuutta mittaavia osa-alueita. Työssä toteutettavan järjestelmän tärkeimpinä laadullisina vaatimuksina voidaan pitää käytettävyyttä, turvallisuutta sekä ylläpidettävyyttä.

Järjestelmän käytettävyyttä suunniteltaessa on otettava huomioon järjestelmän helposti ymmärrettävyys, opittavuus ja helppokäyttöisyys. Järjestelmän täytyy olla myös miellyttävä käyttää. (ISO/IEC 25010 2008, 16.) Työssä toteutettavassa järjestelmässä käytettävyys on syytä ottaa huomioon käyttöliittymää suunniteltaessa. Järjestelmän tulisi olla selkeä ja halutun toiminnon suorittaminen helppoa. Käytettävyyttä voidaan lisätä käyttämällä selkeää värimaailmaa, ja panostamalla järjestelmän visuaaliseen ilmeeseen. Tämän lisäksi järjestelmään voidaan lisätä alue ohjeille, missä kerrotaan, miten eri järjestelmällä tehtävät toiminnot suoritetaan.

Turvallisuuden laadullisena vaatimuksena on ISO/IEC 25010-standardin (2008, 18) mukaan suojata järjestelmä tahattomalta ja tahalliselta murtautumiselta, käytöltä ja sulkemiselta sekä tiedon muuttumiselta ja poistamiselta. Käytännössä tämä tarkoittaa järjestelmän suojaamista potentiaalisilta tietoturvariskeiltä sekä estämällä käyttäjiltä oikeus järjestelmän alueille, joihin käyttäjällä ei ole oikeutta.

Järjestelmän ylläpidettävyydellä tarkoitetaan järjestelmän sopeutumista siihen tehtäviin muutoksiin, muokattavuutta sekä uudelleenkäytettävyyttä (ISO/IEC 25010 2008, 19–20). Työssä toteutettavassa järjestelmässä ylläpidettävyys otetaan huomioon ohjelmoitaessa. Useasti muokattavat ja muutoksien kohteena olevat osat ohjelmakoodia toteutetaan niin, että tarvittavat muutokset täytyy tehdä vain kerran. Esimerkiksi järjestelmän metatiedot ja rakenteeseen liittyvät muutokset voivat olla muutoksien kohteena. Uudelleenkäytettävyys otetaan huomioon toteutettu luomalla järjestelmäkohtaiset funktiot omaan tiedostoonsa, jolloin samaa ohjelmarunkoa voidaan käyttää monen erityyppisen järjestelmän alustana. Kuviossa 5 on esitetty ISO/IEC 25010-standardin mukaiset laadun mittausalueet.



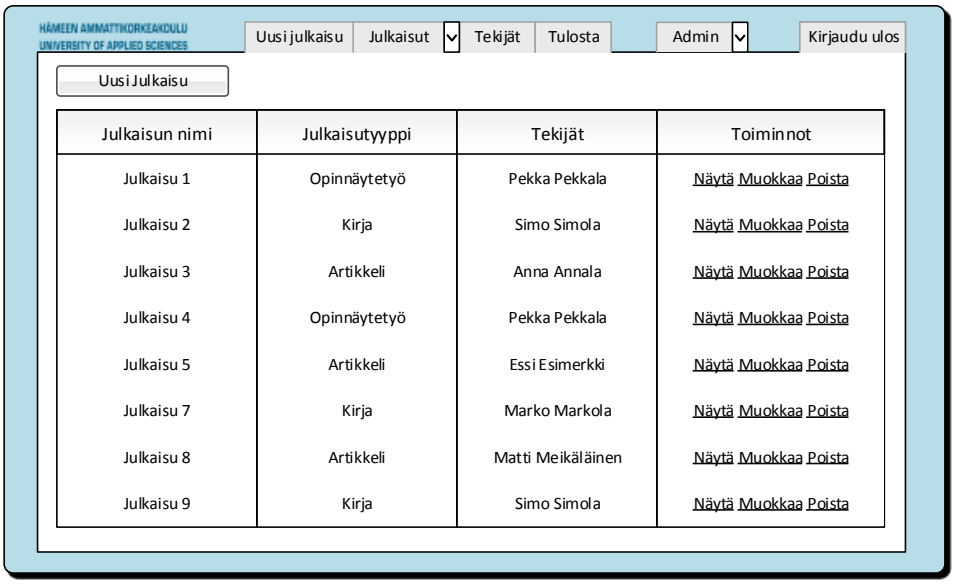
Kuvio 5. ISO/IEC 25010-standardin mukaiset laadun mittausalueet (ISO/IEC 25010 2008, 14)

5.3 Käyttöliittymä

Käyttöliittymällä tarkoitetaan käyttäjän ja ohjelmakoodin välillä käytettävää kommunikointivälinettä. Käyttöliittymä voidaan ajatella siltana, jota pitkin käyttäjän käskyt kulkevat toiminnallisuuden toteuttavalle puolelle. Käyttöliittymä on tärkeä suunnitella hyvin, sillä käytännöllinen ja ulkoasullisesti selkeä käyttöliittymä on loppukäyttäjän näkökulmasta yksi tärkeimmistä asioista, mitä järjestelmältä vaaditaan (Rouhiainen 1997). Käyttöliittymää suunniteltaessa pyrittiin mahdollisimman selkeään lopputulokseen. Ideana oli toteuttaa yksinkertainen käyttöliittymä, jotta loppukäyttäjän näkökulmasta käyttö olisi vaivatonta ja helppoa.

Kuvassa 18 on esitetty käyttöliittymäluonnos, mikä toteutettiin työn suunnitteluvaiheessa. Suunnitellussa käyttöliittymässä julkaisut näytetään kolmessa sarakkeessa ja julkaisuun liittyvät toiminnot yhdessä sarakkeessa. Ajatuksena oli luoda myös linkityksiä tekstikenttiin navigoinnin helpottamiseksi. Linkityksien avulla voitaisiin esimerkiksi tekijän nimeä painamalla nähdä tekijän tiedot sekä tekijän kaikki julkaisut. Toimeksiantajan

logoa painamalla päästään etusivulle, johon voidaan esimerkiksi kirjoittaa ajankohtaista tietoa tai infoa tulevista päivityksistä. Julkaisut-sivua painamalla käyttäjälle avautuu alasvetovalikko, josta julkaisuja voidaan selata julkaisutyypeittäin. Julkaisuista tehtävä CSV-tiedosto luodaan Tulosta-sivulta, jota painamalla käyttäjälle aukeaa sivu, mistä käyttäjä voi valita CSV-tiedostoon tallennettavat julkaisut julkaisuvuoden ja julkaisutyypin mukaan. Admin-sivulta pääkäyttäjä voi hallita käyttäjätunnuksia ja hyväksyä uusia julkaisuja järjestelmään.



Kuva 18. Suunnitteluvaiheessa toteutettu käyttöliittymäluonnos

5.4 Tietokanta

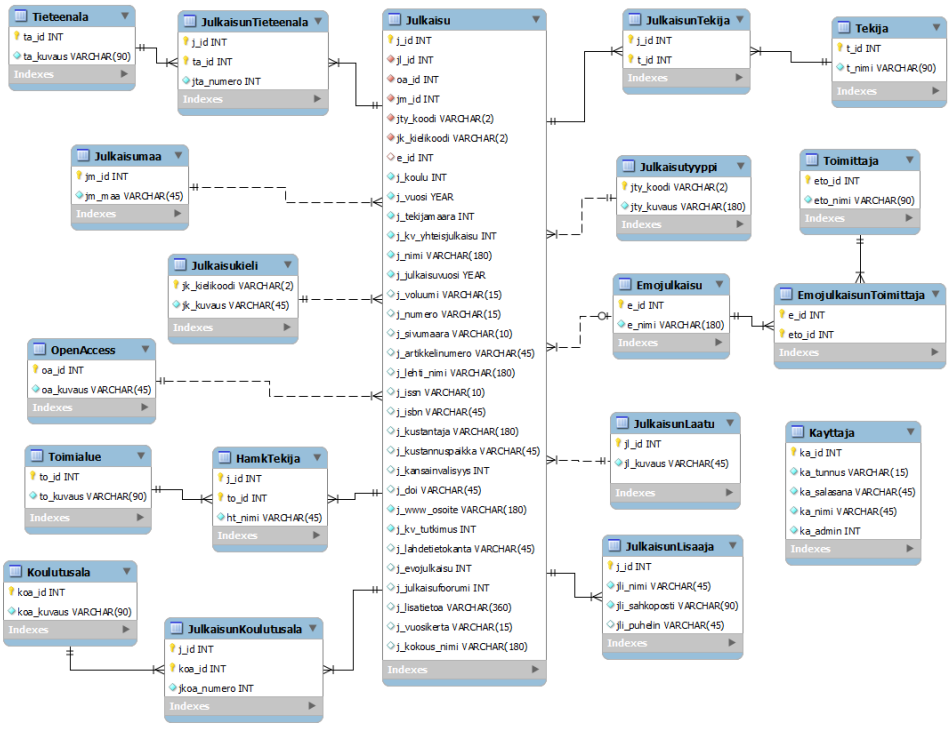
Tietokannan suunnitteleminen oli yksi tärkeimmistä vaiheista järjestelmän suunnittelu- ja toteuttamisprosessin aikana. Hyvin suunniteltu tietokanta lisää järjestelmän käytettävyyttä sekä huolehtii tiedon tallentamisesta oikeassa muodossa. Järjestelmän virheettömän toimimisen kannalta tietokannan suunnitteluprosessissa täytyi ottaa huomioon järjestelmälle asetetut toiminnalliset vaatimukset ja toiveet.

Työssä toteutetun järjestelmän tietokantaa suunnitellessa tietokanta normalisoitiin ensiksi kaikkia kolmea normalisoinnin muotoa noudattaen, mutta myöhemmin ilmeni, että käytettävyyden kannalta tämä tulisi hyvin raskaaksi vaihtoehdoksi toteuttaa tietokanta, sillä tietokantatauluja tulisi todella paljon. Järjestelmän toiminnallisena vaatimuksena oli julkaisutietojen tallentaminen CSV-tiedostoon, mutta mikäli tietokanta olisi suunniteltu käyttämään kaikkia kolmea normaalimuotoa, tiedoston luomiseksi jouduttaisiin tekemään monimutkaisia JOIN-lausekkeita. Monimutkaisten JOIN-lausekkeiden käyttämisen sijaan tietokanta päädyttiin denormalisoimaan toiseen normaalimuotoon. Toisen normaalimuodon käyttäminen ehkäisi tiedon toistumisen, mutta mahdollisti tietokantataulujen yhdistämisen, sillä kaikkien sarakkeiden ei enää tarvinnut olla riippuvaisia taulun pääavaimesta.

Tietokantaa suunniteltaessa yksi keskeisimmistä kysymyksistä oli, miten käsitellä erilaisia julkaisutyyppejä. Julkaisutyypillä tarkoitetaan julkaisun muotoa, joka voi olla muun muassa artikkeli, kirja ja opinnäytetyö. Jokaisella julkaisutyypillä on tyyppikohtaiset metatiedot. Tietokantaa suunniteltaessa tyyppikohtaiset metatiedot täytyi ottaa huomioon, ja lopulta päädyttiin ratkaisuun, jossa luodaan yksi laajempi julkaisu-taulu, mihin kaikkien julkaisujen metatiedot tallennetaan. Kun tietoja tallennetaan tauluun, ohjelmakoodissa valitaan julkaisutyypin mukaan tallennettavat tiedot.

Kuviossa 6 esitetyssä relaatiokaaviossa on havainnollistettu järjestelmälle suunniteltu tietokanta. Tietokannassa suunniteltiin tieteenaloille, koulutusaloille sekä julkaisun, emojulkaisun ja toimeksiantajan organisaatioon kuuluville tekijöille monen suhde moneen yhteystaulut. Yhteystaulut mahdollistivat monen tieteenalan, koulutusalan sekä tekijän yhdistämisen yhteen julkaisuun. Tämän lisäksi opetus- ja kulttuuriministeriön etukäteen määrittelevät luokitukset lisättiin omaan tauluunsa, jotta niille voitaisiin ohjelmakoodissa tehdä pudotusvalikot. Käyttäjille luotiin myös oma taulunsa, johon käyttäjän tunnukset, salasanat ja käyttöoikeudet tallennetaan.

Tietokannan viite-eheydestä huolehditaan käyttämällä viite-eheydestä huolehtivia sääntöjä. Tietokantaan suunniteltiin myös eri julkaisutyypeille omat näkymänsä, jolloin järjestelmässä voidaan listata julkaisuja julkaisutyypeittäin. Näkymien käyttäminen ehkäisee toistuvien hakulausekkeiden käyttämisen ja täten järjestelmän turvallisuutta (Saarelainen 2008, 1–3a).



Kuvio 6. Julkaisujärjestelmän relaatiokaavio

6 TOTEUTUS

Tässä luvussa on käsitelty järjestelmän toiminnallisuuden kannalta keskeisiä toimintoja, tietoturvaratkaisuja sekä havainnollistettu toteutettua käyttöliittymää. Työn toteutuksessa käytettyjen tekniikoiden valitsemiseen vaikutti työn tekijän oma kokemus kyseisistä tekniikoista, sekä niiden yhteensopivuus. Työn toteuttajalle ei ollut ennestään paljoa kokemusta PHP:llä ohjelmoimisesta, mutta sen sijaan käytännön kokemusta MySQL-tietokannoista. Tämä mahdollisti uuden oppimisen sekä vanhan taidon hyödyksi käyttämisen työtä tehdessä.

PHP valikoitu ohjelmakieleksi, sillä se soveltuu hyvin tietokantapohjaisten web-sovellusten ohjelmointikieleksi. Tietokantayhteys on toteutettu käyttäen PHP:n tukemaa PDO-rajapintaa, sillä abstraktitason toteutuksen ansiosta PDO:ta voidaan käyttää melkein kaikkien tietokantaohjelmien kanssa (Wurzer 2012). PHP sopii hyvin käytettäväksi myös HTML:n kanssa, sillä se voi käyttää merkintäkielenään HTML-kieltä ja HTML:n avulla PHP:lle voidaan lähettää tietoa käsiteltäväksi.

Järjestelmän käyttöliittymä on toteutettu Twitter Bootstrap-sovelluskehyksen avulla ja sivuston graafiseen muotoiluun käytetään CSS- ja CSS3-tyyliohjekieliä. Dynaamisen toiminnallisuuden lisäämiseen on työssä toteutetussa järjestelmässä käytetty JavaScriptin jQuery-kirjastoa.

Järjestelmän tiedostorakenne on toteutettu niin, että toiminnallisuuden, tietoturvan ja tietokantayhteyden toteuttavat PHP-funktiot ovat omassa kansiossa. Kuville, CSS-tyyliohjeille sekä JavaScriptille on luotu omat kansionsa, kun taas järjestelmän sivustorakenteen määrittelevät tiedostot on luotu järjestelmän juurikansioon.

6.1 Tietokantayhteys

Tietokantaan yhdistäminen on kriittinen osa järjestelmän toiminnallisuutta. Ilman avointa yhteyttä tietokantaan ei tiedon kirjoittaminen ja tallentaminen onnistu. Mikäli tietokantaan yhdistäminen on toteutettu huonosti, on se suuri tietoturvariski.

Esimerkeissä 12–15 on esitetty Database-luokka, jossa tietokantaan luodaan yhteys PDO:ta käyttäen. Luokka koostuu kolmesta tietokantafunktiosta, joilla tietokantaa käytetään. Ensimmäiseksi luokassa käytettäviin muuttujiin tallennetaan tietokannan nimi, osoite, käyttäjänimi sekä salasana. Tämän lisäksi cont-muuttujan arvoksi alustetaan tyhjä arvo.

```
class Database {

    private static $dbName = 'tietokanta' ;
    private static $dbHost = 'osoite' ;
    private static $dbUsername = 'kayttajanimi';
    private static $dbUserPassword = 'salasana';
    private static $cont  = null;
```

```
public function __construct() {}
```

Esimerkki 12. Database-luokka, jonka avulla tietokantaan yhdistetään PDO:ta käyttäen

Yhdistäminen tietokantaan tapahtuu connect-funktiolla, jossa cont-muuttujan arvoksi asetetaan PDO:ta käyttävä lauseke. Tämä lauseke yhdistää luokan alussa määritetyt muuttujat PDO-olioon, jota käytetään myöhemmin tietokannan käyttämiseen.

```
public static function connect(){
        if ( null == self::$cont ){
        try {
        self::$cont =  new PDO(
        "mysql:host=".self::$dbHost.";"."dbname=".self::$
        dbName, self::$dbUsername,
        self::$dbUserPassword);
        }
        catch(PDOException $e) {
        die($e->getMessage());
        }
        }
        return self::$cont;
}
```

Esimerkki 13. Database-luokka, jonka avulla tietokantaan yhdistetään PDO:ta käyttäen

Tietokantaa käytettäessä käytetään prepareDB-funktiota, johon tuodaan argumenttina sql-muuttuja. Tähän muuttujaan on tallennettu SQL-lauseke jolla halutut CRUD-komennot suoritetaan tietokantaan. PDO tarjoaa prepare toiminnon, jonka avulla erikoismerkit saadaan poistettua SQL-lausekkeista ennen tiedon tallentamista tietokantaan ja näin ollen estetään SQL-injektio.

```
public function prepareDB($sql){
        try {
        $result = self::$cont->prepare($sql);
        return $result;
        }
        catch (PDOException $i) {
        die($i->getMessage());
        }
```

Esimerkki 14. prepareDB-funktio, jolla erikoismerkit poistetaan SQL-lausekkeesta

Tietokantayhteys suljetaan disconnect-funktiolla, mikä tyhjentää cont-muuttujassa sijaitsevan yhdistämiseen käytettävän lausekkeen ja näin ollen estää yhteyden käyttämisen.

```
public static function disconnect(){
        self::$cont = null;
}
}
```

Esimerkki 15. disconnect-funktio, jolla tietokantayhteys suljetaan

6.2 Lomaketietojen suojaaminen

Lomakkeilla voidaan lähettää erilaisia tietoja web-sivujen välillä. Tiedot lähetetään PHP:ssä lomakkeelta POST- ja GET-metodeilla. Käytännössä näiden lähetystapojen erona on se, että GET-metodia käyttäessä tietoa haetaan sivustolta, ja haettava tieto tulee näkyviin sivuston osoiteriville, kun taas POST-metodia käytettäessä tietoa lähetetään sivulle sivupyyntönä, jolloin lomakkeelta lähetetty tieto ei näy käyttäjälle osoiterivillä. (OAMK.)

Koska POST-metodilla voidaan lähettää tietoja web-sivujen välillä, on olemassa potentiaalinen tietoturvauhka. Mikäli käyttäjällä on avoin istunto järjestelmään ja käyttäjä vierailee toisella websivustolla, on olemassa mahdollisuus, että järjestelmään voidaan lähettää tietoja toiselta web-sivustolta POST-metodilla, esimerkiksi AJAX-tekniikan avulla. Kun käyttäjä on kirjautunut järjestelmään, pystytään toiselta sivustolta lähettämään tietoa lomakkeilla, jotka olisivat muuten käytettävissä vain kirjautuneille käyttäjille. (Bulten 2009.)

Tietoturvauhka voidaan ratkaista luomalla jokaiselle lomakkeelle yksilöllinen avain, joka kostuu sattumanvaraisesta numerosta ja käyttäjän IP-osoitteesta. Näiden kahden tiedon yhdistelmä salataan MD5-tiivisteellä. Tämä avain tallennetaan istuntoon, joka mahdollistaa lomakkeelta saapuvan avaimen vertaamisen istunnon avaimeen. Tällä metodilla suojataan järjestelmä Cross-site request forgery-haavoittuvuudelta, kun lomakkeiden tietoja lähetetään.

Esimerkissä 16–19 on esitetty FormKey-luokka, jonka avulla lomakkeille luodaan yksilöllinen avain, jolla estetään POST-metodin väärinkäyttö ja tietojen lähettäminen toisilta websivustoilta. Luokalle luodaan ensimmäiseksi kaksi muuttujaa, joihin uusi ja vanha salausavain tallennetaan. Tämän jälkeen luokalle luodaan rakentaja, jonka sisällä istunnosta haetaan avain, mikäli se on jo olemassa ja tallennetaan saatu arvo old_formKey-muuttujaan.

```
class FormKey{

    private $formKey;
    private $old_formKey;

    function __construct(){
          if(isset($_SESSION['form_key'])){
          $this->old_formKey = $_SESSION['form_key'];
          }
    }
```

Esimerkki 16. FormKey-luokka jonka avulla estetään Cross-site request forgery-haavoittuvuuden käyttäminen lomakkeiden tietoja lähettäessä

Funktiolla generateKey, haetaan käyttäjän IP-osoite ja luodaan uniqid-muuttuja, johon mt_rand-funktiota käyttämällä luodaan sattumanvarainen numero. Kun numero on luotu, se yhdistetään käyttäjän IP-osoitteeseen ja

tämä salataan MD5-tiivisteellä, jonka jälkeen funktio palauttaa tämän arvon.

```
private function generateKey(){
        $ip = $_SERVER['REMOTE_ADDR'];
        $uniqid = uniqid(mt_rand(), true);
        return md5($ip . $uniqid);
}
```

Esimerkki 17. generateKey-funktio, jolla luodaan yksilöllinen salausavain

Funktiolla outputKey saadaan lomakkeelle tulostettua avain, ajamalla generateKey-funktio ja tallentamalla saatu arvo formKey-muuttujaan. Luotu avain tallennetaan istuntoon sekä tulostetaan lomakkeeseen, josta POST-metodia käyttäen se myöhemmin lähetetään tarkistettavaksi.

```
public function outputKey(){
        $this->formKey = $this->generateKey();
        $_SESSION['form_key'] = $this->formKey;

        echo "<input type='hidden' name='form_key'
        id='form_key' value='".$this->formKey."' />";
}
```

Esimerkki 18. outputKey-funktio, jolla tulostetaan lomakkeelle salausavain

Viimeiseksi luokassa ajetaan validate-funktio, jonka palauttamaa arvoa käytetään tarkistuksena lomakkeita sisältävillä sivuilla POST-metodia käytettäessä. Funktion avulla lomakkeelta lähetetty avain verrataan luokan rakentajaan tallennettuun avaimeen. Lomakkeelta lähetettyä avainta ei voida verrata uuteen luotuun avaimeen, koska generateKey-funktio ylikirjoittaa istuntoon tallennetun avaimen. Tästä syystä luokan rakentajaan on tallennettu vanha avain, jotta lomakkeelta saapuvaa avainta voidaan verrata siihen. Mikäli avaimet ovat samat, funktio palauttaa arvon ’true’.

```
public function validate(){

        if($_POST['form_key'] == $this->old_formKey){
        return true;
        }
        else{
        return false;
        }
}
}
```

Esimerkki 19. validate-funktio, jolla tarkistetaan lomakkeelta saapuva avain

Esimerkeissä 16–19 esitettyä FormKey-luokkaa käyttämällä lomakkeen tiedot on suojattu toisilta verkkosivuilta tulevilta syötteiltä. Luokka estää myös lomakkeiden lähettämisen kahteen kertaan, sillä aina, kun tietoa lähetetään, luodaan uusi yksilöllinen salausavain. Tästä johtuen käyttäjä ei

voi vahingossa lähettää tietoja kahteen kertaan, koska validate-funktio palauttaa tällöin arvon ”false”, ja avaimen tarkistaminen epäonnistuu.

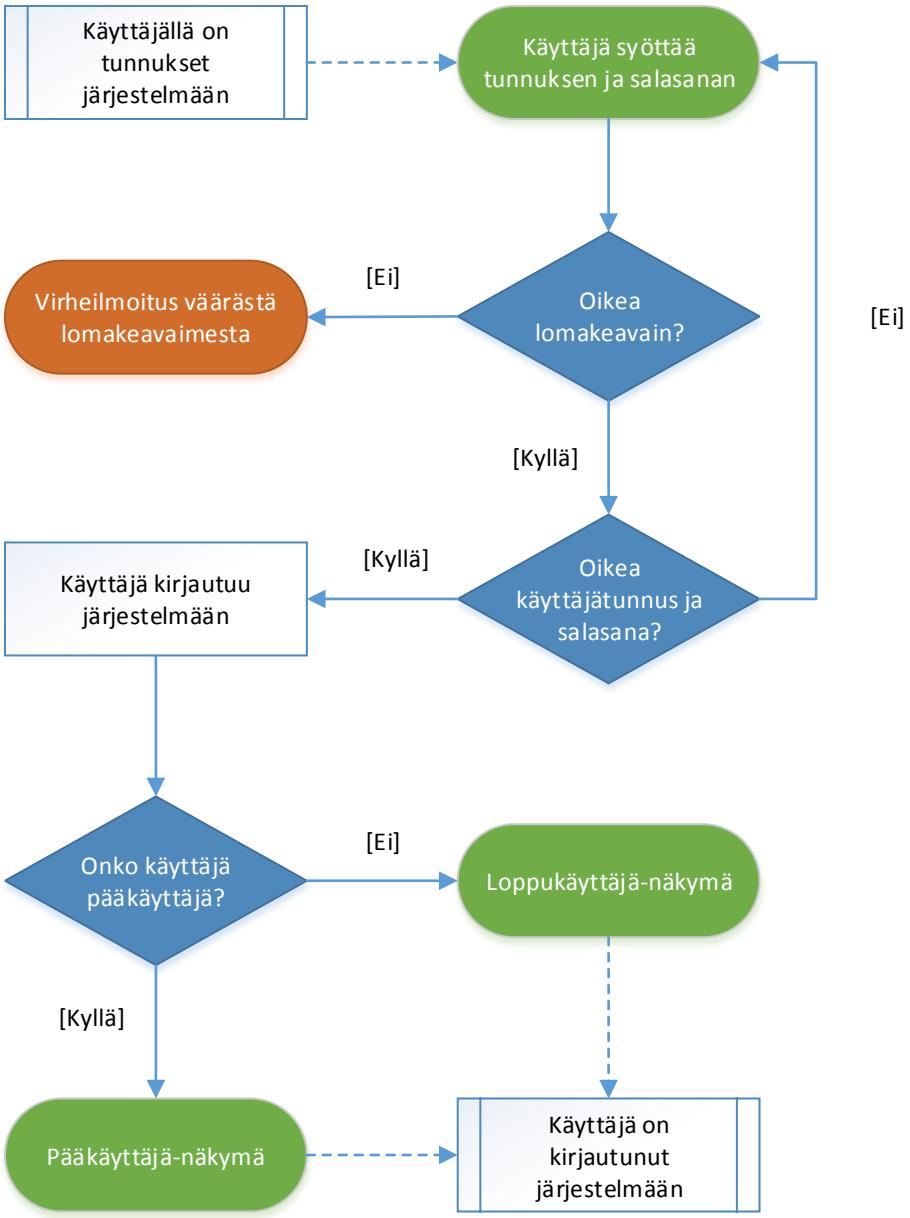
6.3 Käyttäjätunnistus

Järjestelmä, jossa käytetään ominaisuuksia, joihin vain määrätyille henkilöille halutaan antaa käyttöoikeus, on syytä suojata käyttäjätunnistuksella. Käyttäjätunnistuksen tarkoituksena on rajata ulkopuolisten henkilöiden sisäänpääsy järjestelmään ja näin estää tallennetun tiedon väärinkäyttö. (Ballad & Ballad 2009, 95.)

Nykyaikainen lähestymistapa käyttäjätunnistuksen tekemiseen on verrata käyttäjän syöttämää käyttäjätunnusta ja salasanaa tietokannassa sijaitsevaan käyttäjätauluun. Käyttäjätauluun voidaan tallentaa myös käyttäjäkohtaisia tietoja, kuten käyttäjän nimi, yhteystiedot ja taulua voidaan käyttää apuna käyttöoikeuksien yksilöimiseen. Mikäli käyttäjän syöttämät tiedot vastaavat tietokannassa oleviin tietoihin, käyttäjä kirjautuu järjestelmään. (Ballad & Ballad 2009, 114.)

Työssä toteutetussa järjestelmässä tallennetaan tietoja toimeksiantajalla tehdyistä julkaisuista, joten käyttöoikeus halutaan antaa vain henkilöille, jotka käsittelevät näitä tietoja. Käyttäjätunnistus toteutetaan antamalla jokaiselle järjestelmää käyttävälle henkilölle yksilöllinen käyttäjätunnus sekä salasana järjestelmään kirjautumista varten. Käyttäjä voi itse vaikuttaa tietoturvaan valitsemalla salasanan, joka on tarpeeksi pitkä sekä satunnainen.

Kuviossa 7 on esitetty järjestelmässä käytetty kirjautumisprosessi. Kun järjestelmän käyttäjä lähettää kirjautumislomakkeella käyttäjätietonsa, tarkistaa järjestelmä ensin lomakeavaimen oikeellisuuden. Lomakeavain luodaan käyttäjän kirjautumislomakkeelle, kun käyttäjä avaa kirjautumissivun. Tällä varmistetaan että lomake lähetetään alkuperäisestä lähteestä, eikä esimerkiksi toiselta verkkosivustolta CSRF-hyökkäyksenä. Mikäli lomakeavain on oikea, varmistetaan käyttäjätunnus ja salasana tietokannasta. Salasanat on tallennettu tietokantaan tämän työn luvussa 4.2 esitettyä suolaus-tekniikkaa käyttäen. Mikäli käyttäjätunnus ja salasana ovat oikeat, annetaan käyttäjälle pääsy järjestelmään. Tämän jälkeen järjestelmä tarkistaa käyttäjän käyttöoikeudet ja ohjaa käyttäjän oikeaan näkymään.



Kuvio 7. Käyttäjätunnistus esitettynä vuokaaviona

Käyttäjän käyttöoikeudet riippuvat tietokantatauluun tallennetusta admin-luvusta. Mikäli tämä luku on yli 0, asetetaan istunnon nimeksi ”admin” ja käyttäjä saa pääkäyttäjänoikeudet. Mikäli arvo on sen sijaan 0, niin istunnon nimeksi asetetaan ”kayttaja”, ja käyttäjä saa loppukäyttäjän oikeudet. Esimerkissä 20 on havainnollistettu, miten käyttöoikeuksien tarkistaminen on toteutettu koodipuolella.

```
if($row->admin > 0){
$_SESSION["admin"]  = utf8_encode($row->nimi);
}

else {
$_SESSION["kayttaja"]  = utf8_encode($row->nimi);
}
```

Esimerkki 20. Istunnon nimen määritteleminen admin-luvun avulla.

Mikäli peruskäyttäjältä halutaan evätä pääsy vain pääkäyttäjätunnuksille tarkoitetuille alueille tai tietokenttiin, voidaan pääsy rajata alla esimerkissä 21 esitetyllä tavalla. Loppukäyttäjillä on käyttöoikeus julkaisujen lisäämiseen, lukemiseen ja päivittämiseen, mutta heillä ei ole oikeuksia hallita käyttäjätunnuksia. Pääkäyttäjällä on CRUD-komentojen lisäksi oikeudet admin-sivulle, jossa pääkäyttäjä voi hallita loppukäyttäjien tunnuksia.

```
if(isset($_SESSION["admin"])){
/* Tämä sisältö näytetään vain pääkäyttäjille */
}
```

Esimerkki 21. Käyttöoikeuksien rajaaminen istunnon nimen perusteella

6.4 CSV-tiedoston luominen

Työn toteutuksen kannalta yksi oleellisimmista asioista oli CSV-tiedoston luominen järjestelmään syötettyjen julkaisujen metatiedoista. CSV-tiedostoon tallennetaan UTF-8 merkistökoodauksella kaikki opetus- ja kulttuuriministeriön vaatimat metatiedot (Liite 1). Järjestelmän toiminnallisena vaatimuksena oli mahdollisuus tulostaa julkaisut vuosittain, sekä julkaisutyypistä riippuen.

Esimerkissä 22–25 on esitetty tulostamiseen käytettävä funktio. Käyttäjän painaessa käyttöliittymästä tulosta-painiketta, järjestelmä lähettää lomakkeelta tietona julkaisuvuoden sekä julkaisutyypin. Funktio hakee lomakkeelta tulevat tiedot ja tallentaa ne vuosi- ja tyyppi-muuttujaan.

```
public function tulosta() {

      $vuosi = $_POST['julkaisuvuosi'];
      $tyyppi = $_POST['julkaisutyyppi'];
```

Esimerkki 22. Lomakkeelta saapuvat tiedot tallennetaan muuttujiin

filename-muuttujalle määritellään CSV-tiedoston nimi, joka muodostuu db_export-etuliittestä, sekä tulostuspäivämäärästä. Tietojen noutamiseen käytettävä SQL-lauseke tallennetaan parametrisoituna sql-muuttujaan, jonka jälkeen PDO:n prepare-metodia käyttäen haku alustetaan execute-metodille suoritettavaksi. Haku alustetaan, koska PDO:n execute-metodi ei pysty käsittelemään parametrisoituja hakuja ilman alustusta. Tämän jälkeen lomakkeelta haetut tiedot sidotaan sql-muuttujassa käytettäviin parametreihin. Lopuksi hakulauseke suoritetaan execute-metodilla.

```
      $filename = "db_export_".date("d_m_Y").".csv";
      $sql = 'SELECT * FROM julkaisu WHERE j_vuosi =
      :julkaisuvuosi AND jty_koodi = :julkaisutyyppi';
      $result = $this->pdo->prepare($sql);
      $result->bindParam(':julkaisuvuosi', $vuosi);
      $result->bindParam(':julkaisutyyppi', $tyyppi);
      $result->execute();
```

Esimerkki 23. Tiedon hakemiseen ja alustamiseen käytettävät PDO-funktiot

Seuraavaksi määritellään tiedoston tallentamiseen käytettävät muotoilut. Tiedosto tallennetaan UTF-8 merkintäkoodauksella, ja tiedostomuodoksi määritellään CSV-muoto. Tämän jälkeen filename-muuttujaan tallennettu nimi asetetaan tiedostonimeksi ja estetään selainta luomasta tiedostosta välimuistitallennetta. Lopuksi tiedostolle annetaan tuki UTF-8 BOM-tavujärjestykselle, jotta Excel tunnistaisi erikoismerkit.

```
header('Content-Encoding: UTF-8');
header('Content-type: text/csv; charset=UTF-8');
header('Content-Disposition: attachment; filename='.
$filename);
header('Pragma: no-cache');
header("Expires: 0");
echo "\xEF\xBB\xBF";
```

Esimerkki 24. CSV-tiedoston kirjoittamiseen käytettävien muotoilujen määritteleminen

Viimeiseksi valitaan tiedostoon tallennettavat tiedot ja avataan tiedosto kirjoittamista varten handle-muuttujaan. Tiedot tallennetaan fputcsv-metodilla, ja ensimmäiseksi tiedostoon talennetaan sarakkeiden nimet. Esimerkin yksinkertaitamiseksi tallennettavat tiedot on esitetty esimerkissä sarake ja tieto-nimillä, oikeiden nimien sijaan. Tämän jälkeen foreach-metodilla käydään lävitse tietokantahaun tuloksia niin monta kertaa kuin rivejä on, ja tallennetaan haetut tiedot tietokannasta CSV-tiedostoon omille rivilleen. Tämän jälkeen tiedoston kirjoittaminen ja tietokantayhteys suljetaan, ja käyttäjä voi ladata tiedoston tietokoneelleen.

```
$handle = fopen("php://output", "w");
fputcsv($handle, array('Sarake 1','Sarake 2','Sarake
3'));

foreach($result as $row){
fputcsv($handle, array($row['tieto1'],$row['tieto2'],
$row['tieto3']));
}
fclose($handle);
$this->database->disconnect();
}
```

Esimerkki 25. CSV-tiedoston tietojen kirjoittaminen fputcsv- ja foreach-metodeilla.

6.5 Käyttöliittymä

Järjestelmän graafinen käyttöliittymä toteutettiin Twitter Bootstrap–sovelluskehyksellä. Twitter Bootstrap-sovelluskehyksen käyttämiseen päädyttiin, koska se tarjosi valmiiksi määritellyt CSS-tyyliohjeet taulukoiden, painikkeiden, lomakkeiden ja elementtien muotoiluun. Järjestelmään kirjautumiseen käytettävä lomake toteutettiin työn toteuttajan määrittelemillä CSS-tyyliohjeilla, jotta lomakkeen ulkoasu saatiin paremmin vastaamaan toimeksiantajan käyttämää visuaalista ilmettä verkkosivuilla.

6.5.1 Järjestelmään kirjautuminen

Järjestelmän kirjautumislomakkeen visuaalinen ilme toteutettiin käyttämällä HTML-merkintäkieltä ja CSS-tyyliohjeita. Kuvassa 19 esitetyn järjestelmään kirjautumiseen käytettävän lomakkeen muotoilussa käytettiin toimeksiantajan logon värejä ja visuaalisuutta lisättiin luomalla varjostuksia CSS3-tyyliohjeilla. Tämän lisäksi kirjautumislomakkeelle annettiin kymmeniä muita tyyliohjeita liittyen tekstin ja kenttien asetteluun sekä HTML-elementtien muotoiluun (Liite 2).

Kuva 19. Julkaisujärjestelmän kirjautumislomake

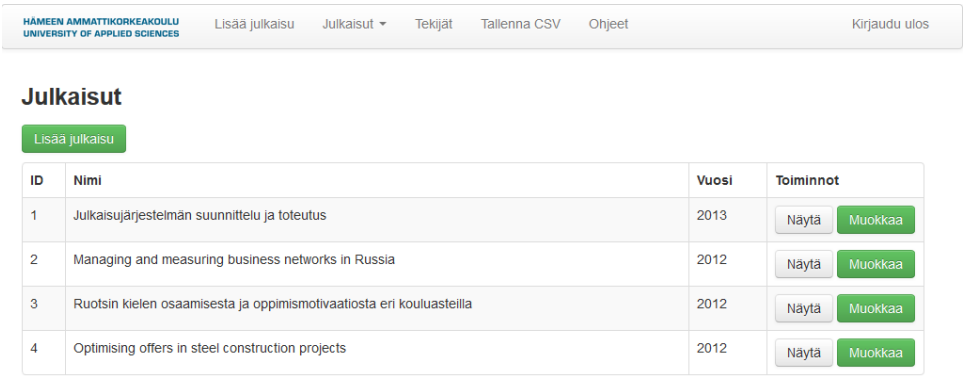
6.5.2 Julkaisun lisääminen järjestelmään

Julkaisun lisääminen järjestelmää on kaksiosainen prosessi. Ensiksi käyttäjä valitsee julkaisun tyypin kuvassa 20 esitetyltä Lisää Julkaisu -sivulta. Tämän jälkeen valitun julkaisutyypin mukaan järjestelmä ohjaa käyttäjän oikealle sivulle käyttäjän painaessa Seuraava-painiketta. Koska jokaisella julkaisutyypillä on eri tiedot, jotka järjestelmään tallennetaan, niin myös erityyppisillä julkaisuilla on omat sivut tietojen lisäämiseen. Mikäli käyttäjä valitsee julkaisun tyypiksi esimerkiksi kirjan, niin käyttäjä ohjataan käyttäjä kirjan lisäämiseen tarkoitetulle sivulle (Liite 3).

Kuva 20. Julkaisutyypin valintaan käytettävä lomake

6.5.3 Loppukäyttäjän näkymä

Kun käyttäjä kirjautuu järjestelmään, käyttäjälle annetaan käyttöoikeudet ja ohjataan oikeaan näkymään. Kuvassa 21 on esitetty loppukäyttäjän näkymä julkaisut-sivulta. Mikäli käyttäjä saa loppukäyttäjän-käyttöoikeudet, hän pystyy lisäämään ja muokkaamaan julkaisuja, mutta ei poistamaan niitä. Pääkäyttäjän oikeuksilla kuvassa 21 esitetyssä näkymässä pystyisi myös poistamaan julkaisuja.

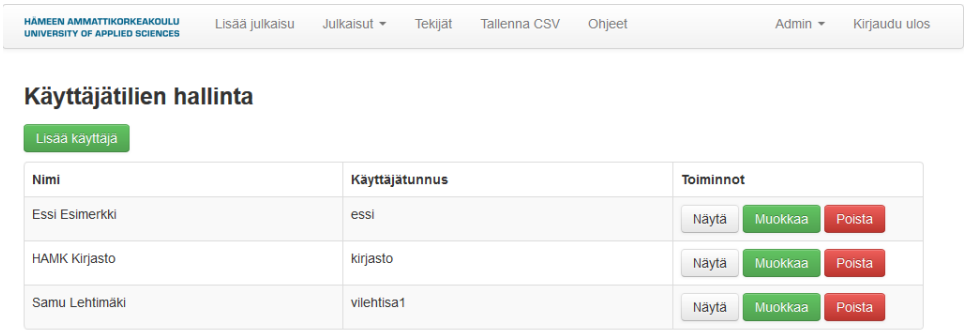


Kuva 21. Loppukäyttäjän näkymä julkaisut-sivulta

6.5.4 Pääkäyttäjän näkymä

Mikäli käyttäjä kirjautuu järjestelmään pääkäyttäjä-tunnuksilla, ohjataan käyttäjä pääkäyttäjä-sivulle, josta voidaan hallita käyttäjätilejä. Kuvassa 22 on esitetty käyttäjätilien hallintapaneeli, missä pääkäyttäjä pystyy lisäämään uusia käyttäjiä, näyttämään käyttäjien tiedot, muokata tietoja ja poistaa käyttäjiä. Käyttäjätilien hallinta on toteutettu, jotta työn toimeksiantajan henkilökunta voi itse lisätä uusia käyttäjätilejä tarvittaessa, ilman että niitä jouduttaisiin lisäämään tietokantaohjelmiston kautta. Tietoturvallisista syistä pääkäyttäjä ei voi luoda tai näyttää muiden pääkäyttäjien tie-

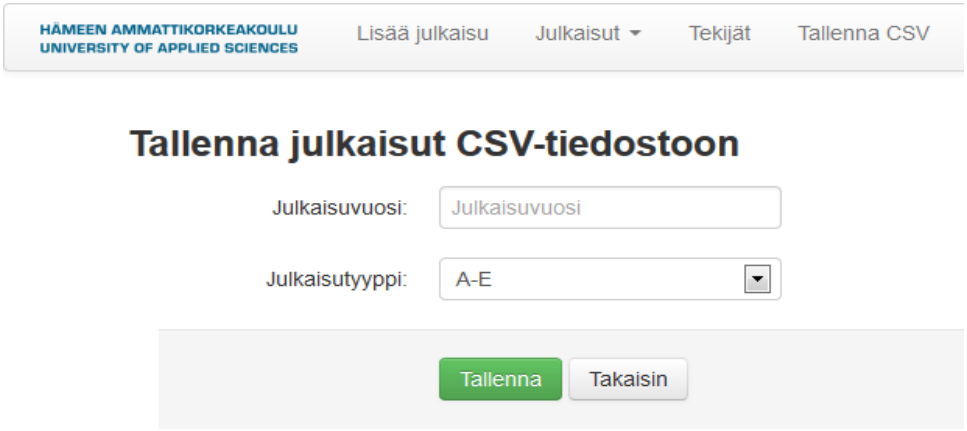
toja, sillä mikäli pääkäyttäjän tunnukset vuotaisivat vääriin käsiin, voisi pääkäyttäjätunnukset omistava henkilö poistaa kaikki järjestelmän muut pääkäyttäjät.



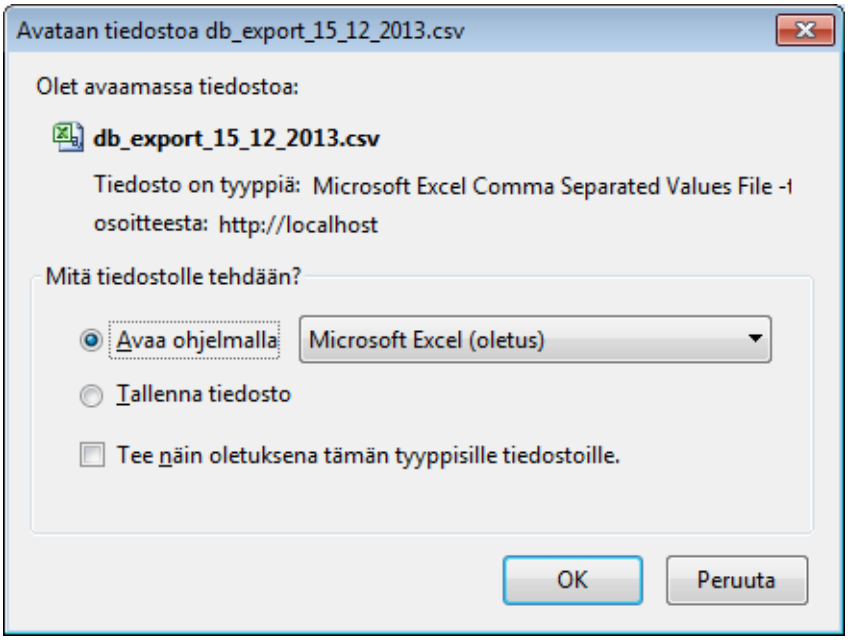
Kuva 22. Käyttäjätilien hallintapaneeli

6.5.5 CSV-tiedoston tallentaminen

CSV-tiedosto tallennusta varten järjestelmään on tehty oma sivu, mistä käyttäjä pystyy tallentamaan järjestelmästä löytyvien julkaisujen metatiedot julkaisuvuoden ja julkaisutyypin mukaan. Opetus- ja kulttuuriministeriö haluaa julkaisujen metatiedot vuosittain kahtena tiedostona, joista toisessa on julkaisutyyppien A-E metatiedot ja toisessa julkaisutyyppien F-I metatiedot. Kuvassa 23 on esitetty tiedoston tallentamiseen käytettävä sivu. Käyttäjän painaessa Tallenna-painiketta, lähetetään käyttäjän syöttämät tiedot järjestelmään, jonka jälkeen käyttäjälle aukeaa kuvassa 24 esitetty ikkuna, josta käyttäjä voi tallentaa tiedoston tietokoneelle tai avata tiedoston esimerkiksi Microsoft Excel-ohjelmistolla.



Kuva 23. CSV-tiedoston tallentamiseen käytettävä sivu



Kuva 24. Ikkuna, josta CSV-tiedosto voidaan tallentaa tai avata ohjelmalla

7 JATKOKEHITYS

Työssä toteutettu järjestelmä on vielä kehitysasteella, joten jatkokehityksen suunnittelu oli olennainen osa työtä. Jatkokehitystä tarvitaan etenkin lomaketietojen virheentarkistuksessa, sekä tietokantataulujen käyttöön liittyvien SQL-lausekkeiden muokkaamisessa.

Järjestelmän rakenne sekä suurin osa funktioista, kuten tietoturvafunktiot ja tietokannan käyttöön liittyvät funktiot on suunniteltu niin, että ne voidaan valjastaa hyvin erilaisiin käyttötarkoituksiin. Järjestelmää voidaan esimerkiksi käyttää alustana erilaisille sisällönhallintajärjestelmille. PDO:n abstraktitason tietokantametodien ja ajureiden ansiosta järjestelmässä voidaan käyttää tietokantaohjelmistona esimerkiksi Microsoft SQL Serveriä, PostgreSQL:ää tai Oraclea. Vaikka järjestelmä otetaan käyttöön aluksi vain sisäverkossa, niin järjestelmässä käytettävät tietoturvaratkaisut mahdollistavat järjestelmän käyttöönottamisen tietoturvallisesti myös ulkoisella palvelimella.

Jatkokehityksen helpottamiseksi ohjelmakoodi kommentoitiin, jotta järjestelmän toiminallisuutta olisi helpompi ymmärtää ja seurata. Järjestelmän eri sivuilla usein toistuvat tiedot, kuten metatiedot sekä valikot on luotu omiin tiedostoihinsa, jotta mahdolliset muutokset tietoihin täytyy kirjoittaa vain kertaalleen. Järjestelmässä käytettävät tyylitiedostot ja jQuery-kirjasto on tallennettu paikallisesti järjestelmän kansioihin, joten järjestelmä toimii myös ilman verkkoyhteyttä.

Järjestelmä ja sen tietokanta on suunniteltu opetus- ja kulttuuriministeriön vuonna 2012 vaatimien metatietojen pohjalta. Vuodesta 2013 lähtien opetus- ja kulttuuriministeriö tulee vaatimaan myös tietoa siitä, onko julkaisut

tehty yhteistyössä jonkin kansallisen tutkimusorganisaation kanssa. Tarkempaa tietoa käytettävistä koodeista ja tiedostomuodoista ei vielä ole.

Järjestelmän tietokanta on suunniteltu tukemaan työn toimeksiantajan toivomaa ominaisuutta kokoelmateoksien ja tekijöiden selaamisen, mutta ohjelmakoodillinen toteutus on vielä tekemättä. Sähköpostien lähettäminen järjestelmästä on myös osa jatkokehityssuunnitelmaa.

8 YHTEENVETO

Tämän työn tarkoituksena oli toteuttaa toimeksiantajalle järjestelmä tietokantoineen, joka korvaisi käytössä olevan virhealttiin ja ajallisesti hitaan tiedonkäsittelyprosessin. Tarkoituksena oli myös toteuttaa toiminto, jolla järjestelmästä saadaan tallennettua opetus- ja kulttuuriministeriön vaatimat metatiedot CSV-muodossa. Tämän lisäksi työn tarkoituksena oli kartoittaa PHP-järjestelmien tietoturvauhkia, sekä suunnitella ja toteuttaa järjestelmä tietoturvallisesti. Jatkokehittäminen oli pidettävä mielessä koko työn suunnittelu- ja toteutusprosessin ajan, jotta järjestelmän jatkokehittäminen olisi vaivatonta ja helppoa tulevaisuudessa.

Suunnitteluvaiheessa toimeksiantajan kanssa kartoitetut toiminnalliset ja laadulliset tavoitteet toimivat pohjana järjestelmän ja tietokannan suunnittelulle. Tietokanta suunniteltiin niin, että sillä pystytään toteuttamaan kaikki toimeksiantajan asettamat tavoitteet ja toiveet. Käyttöliittymää suunniteltaessa pidettiin mielessä järjestelmän loppukäyttäjät ja käyttöliittymästä suunniteltiin selkeä ja helppokäyttöinen. Vaikeinta suunnitteluvaiheessa oli tietokannan toteuttaminen niin, että se palvelisi mahdollisimman hyvin järjestelmän ensisijaista käyttötarkoitusta.

Tietokantaa suunniteltaessa kokeiltiin kolmea erilaista ratkaisuvaihtoehtoa, joista käytettäväksi valikoitui työssä esitetty osittain denormalisoitu tietokantaratkaisu. Myös opetus- ja kulttuuriministeriön vaatimien metatietojen (Liite 1) kartoittaminen ja informaation soveltaminen tietokantaa suunniteltaessa aiheutti alussa ongelmia, sillä eri julkaisutyypeiltä vaadittiin erilaisia metatietoja, joka aiheutti hämmennystä työn alkutaipaleella. Työssä toteutetulle järjestelmälle saatiin kuitenkin onnistuneesti suunniteltua tehokas ja tavoitteet täyttävä tietokantaratkaisu, joka sisältää tietokantataulut myös järjestelmään jatkokehitteillä oleville ominaisuuksille.

Työssä kartoitettiin myös PHP-järjestelmien potentiaalisia tietoturvauhkia, joista suurimmiksi uhkiksi ilmenivät käyttäjän syötteeseen liittyvät SQL-injektiot sekä CSRF- ja XSS-hyökkäykset. Vaikka järjestelmä oli suunniteltu käytettäväksi sisäverkossa, oli jatkokehityksen kannalta tärkeää suojata järjestelmä valmiiksi potentiaalisilta tietoturvauhkilta, mikäli järjestelmä otettaisiin joskus käyttöön ulkoisessa verkossa. SQL-injektion estämiseen käytettiin PHP:n PDO-kirjaston tarjoamaa funktiota SQL-lausekkeiden valmistelulle, sekä parametrisoituja hakulausekkeita. XSS-hyökkäykset estettiin käyttämällä PHP:n htmlspecialchars-funktiota, joka poistaa erikoismerkit käyttäjän syötteestä. Selainpohjaiselta CSRF-

hyökkäykseltä suojauduttiin käyttämällä lomakkeissa lomakeavaimia. Tämän lisäksi salasanojen suojaamisessa käytettiin SHA-1- ja MD5-tiivisteitä sekä suolaus-tekniikkaa.

Työn toteutusvaiheessa ohjelmoitiin hankitun tiedon pohjalta käyttöliittymä sekä tietokannan käyttämiseen käytettävät funktiot. Tämän lisäksi ohjelmoitiin tietoturvafunktiot sekä CSV-tiedoston luomiseen käytettävät funktiot. Loppukäyttäjille ja pääkäyttäjille luotiin käyttöoikeusluokat, joiden perusteella yksilöitiin käyttöliittymänäkymiä. Pääkäyttäjille luotiin admin-sivu, josta pääkäyttäjät voivat luoda uusia käyttäjätilejä, sekä muokata ja poistaa loppukäyttäjien tunnuksia. Toteutusvaiheessa ohjelmointiin kului reilusti enemmän aikaa, mitä työn toteuttaja osasi ennalta odottaa, minkä takia osa järjestelmässä käytettävistä funktioista vaatii vielä jatkokehittämistä. Toteutusvaiheessa saatiin kuitenkin luotua toimiva runko, jonka ympärille on helppo lähteä jatkokehittämään järjestelmää.

Työn toteuttajalle ei ollut ennestään paljoa kokemusta PHP-kielestä ja järjestelmien suunnittelemisesta ja toteuttamisesta, joten opinnäytetyö tarjosi hyvät puitteet uuden oppimiselle. Alussa ohjelmoiminen tuotti ongelmia, mutta työn toteuttamisen aikana ohjelmointitaidot kehittyivät ja työn loppupäässä ohjelmoiminen sujui ongelmitta. Kaikista vaikeimmaksi osoittautui tietokannan suunnitteleminen ja toteuttaminen niin, että se palvelisi mahdollisimman hyvin järjestelmän käyttötarkoitusta.

Työssä toteutettu järjestelmä on vielä kehitysasteella, joten jatkokehittäminen oli olennainen osa työtä. Jatkokehittämisen helpottamiseksi ohjelmakoodi kommentoitiin ja ohjelmakoodissa usein toistuvat tiedot sijoitettiin omiin tiedostoihinsa, jotta muutokset tarvitsisi kirjoittaa vain kertaalleen. Jatkokehittämistä tarvitaan etenkin SQL-lausekkeiden soveltamisessa tietokannan käyttämistä varten eri tilanteissa, sekä lomaketietojen virheiden tarkistamisessa. Tämän lisäksi emojulkaisujen ja tekijöiden selaamisen mahdollistaminen on osa jatkokehitystä. Työn toimeksiantajan toiveena oli myös sähköpostipalvelun luominen osaksi järjestelmää, joten sen kehittäminen on yksi tulevaisuuden haasteista.

Tietolähteinä työssä käytettiin toimeksiantajan kanssa käytyjä keskusteluita ja opinnäytetyölle luotua Wikiä sekä PHP- ja MySQL-aiheisia kirjoja. Tämän lisäksi lähteinä käytettiin toimeksiantajan palveluista kertovaa mainosesitettä ja ISO/IEC-standardin dokumenttia sekä erilaisia verkkotekstejä ja opintomateriaalia. Tiedon löytäminen ei tuottanut ongelmaa, vaan ongelmaksi muodostuikin tiedon liika löytyminen. Lähteitä valittaessa kiinnitettiin huomiota etenkin kirjoittajan ammattitaitoon sekä lähteenä toimivan sivuston luotettavuuteen.

Työn tuloksena syntyi toimiva ja tietoturvallinen runko tietokantoineen julkaisujärjestelmälle, joka tulee parantamaan pienellä jatkokehittämisellä tällä hetkellä käytössä olevaa tiedonkäsittelyprosessia. Toimeksiantajalle vaikeuksia aiheuttanut CSV-tiedoston luominen saatiin toimimaan järjestelmässä, kuten myös tietokannan käyttämiseen liittyvät funktiot.

LÄHTEET

Atwood, J. 2008. Maybe Normalizing Isn't Normal.
http://www.codinghorror.com/blog/2008/07/maybe-normalizing-isnt-normal.html
Viitattu 25.11.2013

Ballad, T. Ballad, W. 2009. Securing PHP Web Applications. Pearson Education, Inc.

Bookwalter, J. 2011. What Is CSS3?
http://www.maclife.com/article/features/what_css3
Viitattu 09.12.2013

Bulten, W. 2009. Secure Your Forms With Form Keys.
http://net.tutsplus.com/tutorials/php/secure-your-forms-with-form-keys/
Viitattu 19.11.2013

Cochran, D. 2012. Twitter Bootstrap 101 – Introduction.
http://webdesign.tutsplus.com/tutorials/complete-websites/twitter-bootstrap-101-introduction/
Viitattu 10.12.2013

Doyle, M. 2010. Beginning PHP 5.3. Wiley Publishing, Inc.

Downs, K. 2008. Denormalization Patterns.
http://database-programmer.blogspot.fi/2008/04/denormalization-patterns.html
Viitattu 25.11.2013

Ekonoja, A. Lahtonen, T. Mäntylä, J. 2004. Relaatiotietokantojen peruskäsitteet.
http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index2.html
Viitattu 22.11.2013

Falck, K. 2008. jQuery mullistaa JavaScriptin.
http://kfalck.net/2008/04/29/jquery-mullistaa-javascriptin
Viitattu 26.11.2013

Huotari, J. 2012. SQL-kielen perusteet.
http://homes.jamk.fi/~huojo/opetus/IIZO3030/SQLopas.pdf
Viitattu 10.12.2013

Hämeen ammattikorkeakoulu. 2013. Kirjasto- ja tietopalvelut, mainosesite.
Viitattu 22.11.2013

Immonen, J. 2002. Johdatus ohjelmistotuotantoon, luentomoniste.
http://cs.joensuu.fi/~jimmonen/jot_moniste/jot_moniste_121.html
Viitattu 25.11.2013

ISO/IEC 25010. 2008. Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Software and quality in use models.
http://sa.inceptum.eu/sites/sa.inceptum.eu/files/Content/ISO_25010.pdf
Viitattu 08.12.2013

Järvinen, P. Järvinen, A. 2000. Tutkimustyön metodeista. Opinpajan kirja, Tampere.

Kuivanen, I. 2005. Peruskäyttäjän tietoturvaa.
http://cs.stadia.fi/~kuivanen/tietoturva/
Viitattu 22.11.2013

Laaksonen, A. 2010. Tietoturva, SQL-injektio.
http://www.cs.helsinki.fi/u/ahslaaks/tsoha/turva.html
Viitattu 22.11.2013

Laaksonen, A. 2011. PHP-ohjelmointi.
http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_01
Viitattu 25.11.2013

Murach, J. Harris, R. 2010. Murach's PHP and MySQL. Mike Murach & Associates, Inc.

Nixon, R. 2012. Learning PHP, MySQL, JavaScript and CSS, Second Edition. O'Reilly Media, Inc.

OAMK. Web-sovellusten ohjelmointi: Selaimen ja Web-palvelimen välinen yhteys.
http://www.oamk.fi/sbc/www/http.php
Viitattu 19.11.2013

Opetus- ja Kulttuuriministeriö. 2012. Ammattikorkeakoulujen tiedonkeruukäsikirja.
http://tietomalli.csc.fi/muut_tiedostot/OKM_ammattikorkeakoulut_2012.pdf
Viitattu 03.12.2013

Podila, P. 2013. HTTP: The Protocol Every Web Developer Must Know – Part 1.
http://net.tutsplus.com/tutorials/tools-and-tips/http-the-protocol-every-web-developer-must-know-part-1/
Viitattu 02.12.2013

Pulkkinen, M. 2012. Tietokannan normalisoinnista voi tinkiä – jos tietää mitä siitä seuraa.
http://www.siili.fi/fi/asiantuntijuus/tietokannan-normalisoinnista-voi-tinki%C3%A4-%E2%80%93-jos-tiet%C3%A4%C3%A4-mit%C3%A4-siit%C3%A4-seuraa
Viitattu 25.11.2013

Rouhiainen, E. 1997. Käyttöliittymän visuaalinen suunnittelu. Ohjelmistotekniikan seminaarityö.
http://www.mit.jyu.fi/opiskelu/seminaarit/bak/kayttoliittyma/index.html
Viitattu 05.12.2013

Saarelainen, T. 2008a. Näkymät ja hakemistot.
http://www2.kyamk.fi/~atesa/db/Tietokannat2008_nakymat_hakemistot_ja_valtuudet.pdf
Viitattu 17.10.2013

Saarelainen, T. 2008b. Tietokannan normalisointi ja normaalimuodot.
http://www2.kyamk.fi/~atesa/db/Tietokannat2008_normaalimuodot.pdf
Viitattu 17.10.2013

Shiflett, C. 2006. Essential PHP Security. O'Reilly Media, Inc.

Snyder, C. Myer, T. Southwell, M. 2010. Pro PHP Security From Application Security Principles to the Implementation of XSS Defenses, Second Edition. Apress.
Viitattu 25.11.2013

Tatroe, K. MacIntyre, P. Lerdorf, R. 2013. Programming PHP, Third Edition. O'Reilly Media, Inc.

The jQuery Foundation. 2013. What Is jQuery?
http://jquery.com
Viitattu 26.11.2013

The PHP Group. 2013. PHP Manual, What Do I Need?
http://php.net/manual/en/tutorial.requirements.php
Viitattu 15.10.2013,

Viestintäviraston tietoturvayksikkö CERT-FI. 2013. Varoitus julkaistu: tietomurtojen sarjassa varastettu suomalaisten tietoja.
http://www.cert.fi/tietoturvanyt/2013/09/ttn201309131451.html
Viitattu 22.11.2013

W3Schools. HTML5 Introduction.
http://www.w3schools.com/html/html5_intro.asp
Viitattu 09.12.2013

Wurzer, E. 2012. Why you Should be using PHP's PDO for Database Access.
http://net.tutsplus.com/tutorials/php/why-you-should-be-using-phps-pdo-for-database-access/
Viitattu 26.11.2013

Zeller, B. 2008. Popular Websites Vulnerable to Cross-Site Request Forgery Attacks.
https://freedom-to-tinker.com/blog/wzeller/popular-websites-vulnerable-cross-site-request-forgery-attacks/
Viitattu 11.12.2013

Liite 1

OPETUS- JA KULTTUURIMINISTERIÖN VAATIMAT METATIEDOT

Metatieto	**Tiedon muoto**
1. Julkaisutyyppi	Julkaisutiedonkeruun käsikirjan mukainen **koodi**
2. Tieteenala	Yhdellä julkaisulla on vähintään 1 ja enintään 6 **tieteenalakoodia**. *Tilastokeskuksen tieteenala 2010 -luokituksen mukainen 3- tai 4-numeroinen arvo (esim. 212). Julkaisutyypeissä F, H ja I erittelemätön tieteenalakoodi (NNN)*
3. Julkaisun koulutusala	Yhdellä julkaisulla on vähintään 1 ja enintään 6 **koulutusalakoodia**. *Opetushallinnon koulutusluokituksen 2002 mukainen 1-numeroinen koodi (esim. 2). Julkaisutyypeissä F, H ja I voidaan käyttää erittelemätöntä koulutusalakoodia (N).*
4. Organisaation tekijät	Tekijätietojen esitysmuodon täytyy olla yhdenmukainen korkeakoulukohtaisesti ja nimien erottimena täytyy käyttää puolipistettä. Etunimi ilmoitetaan ensisijaisesti auki kirjoitettuna. Tekijätiedot ilmoitetaan ensisijaisesti muodossa ”von Hummel, Essi; Möttönen, Matti”. Huom. Mikäli kyseisen kentän tiedot sisältävät puolipisteitä (julkaisulla enemmän kuin yksi tekijä), kentän alkuun ja loppuun lisätään lainausmerkit.
5. Organisaation alayksikkö (vapaaehtoinen)	Vapaamuotoinen tekstikenttä. Organisaation alayksiköt erotetaan toisistaan puolipisteillä.
6. Julkaisun tekijät	Julkaisun täydelliset tekijätiedot (ml. ulkomaiset tekijät) siinä muodossa ja järjestyksessä, jossa ne on listattu alkuperäisessä julkaisussa. Mikäli kyseisen kentän tiedot sisältävät puolipisteitä (julkaisulla enemmän kuin yksi tekijä), kentän alkuun ja loppuun lisätään lainausmerkit.
7. Julkaisun tekijöiden lukumäärä	Jos julkaisussa on tekijöitä enemmän kuin 20 ilmoitetaan tekijöiden kokonaislukumäärä. Jos tekijöitä on vähemmän kuin 20, kenttä on vapaaehtoinen. Tieto kokonaislukuna.
8. Kansainvälinen yhteisjulkaisu	**Koodi**: 0 - ei ole kv-yhteisjulkaisu tai 1 - on kv-yhteisjulkaisu *Kuvaus: Kansainvälinen yhteisjulkaisu tarkoittaa, että tekijöissä on vähintään yksi muun kuin suomalaisen organisaation palveluksessa oleva henkilö. Jos henkilö on palvelussuhteessa sekä suomalaiseen että ulkomaalaiseen organisaatioon ja on merkinnyt julkaisuun affiliaatiokseen molemmat, katsotaan julkaisu kansainväliseksi yhteisjulkaisuksi.*

9. Julkaisun nimi	Julkaisun nimi siten kun se on artikkelissa tai teoksessa mainittu. Vapaamuotoinen tekstikenttä. Pääotsikko ja mahdollinen alaotsikko tallennetaan samaan kenttään, ja ne voi erottaa toisistaan välilyönti, kaksoispiste, välilyönti -yhdistelmällä. Mikäli kyseisen kentän tiedot sisältävät puolipisteitä, kentän alkuun ja loppuun lisätään lainausmerkit.
10. Julkaisuvuosi	Nelinumeroinen vuoden esitysmuoto esim. 2007.
11. Volyymi	Vapaamuotoinen tekstikenttä.
12. Numero	Vapaamuotoinen tekstikenttä.
13. Sivut	Tieto kuten viitetiedoissa on esitetty, esim. 1-20.
14. Artikkelinumero	Vapaamuotoinen tekstikenttä. Osa tieteellisistä aikakauslehdistä käyttää artikkelinumeroa. Tämä ilmoitetaan siinä muodossa kun se on julkaisussa. Koskee julkaisutyyppejä A1, A2, A4, B1, B3.
15. Julkaisun kieli	Tilastokeskuksen kielet 2003 (http://www.stat.fi/meta/luokitukset/kieli/001-2003/index.html)-**luokituksen mukainen 2-kirjaiminen tai 2-numeroinen arvo**, esim. fi.
16. Lehden/sarjan nimi	Vapaamuotoinen tekstikenttä. Lehden/sarjan nimi mahdollisimman täydellisenä (kokonaan auki kirjoitettuna, ei lyhenteitä). Koskee julkaisutyyppejä A1, A2, A4, B1, B3, D1, D3, E1.
17. ISSN	2 kertaa 4 merkkiä väliviivalla ilman ylimääräisiä merkkejä, esim. 0090-8258.
18. ISBN	Julkaisun tai emojulkaisun ISBN-numero. Koskee julkaisutyyppejä A3, B2, C1, C2, D2, D5, E2. Puuttuva tieto merkitään koodilla 9999.
19. Emojulkaisun nimi	Vapaamuotoinen tekstikenttä. Emojulkaisun (esim. artikkelikokoelman) nimi. Koskee julkaisutyyppejä A3, B2, C2, D2, D5.
20. Emojulkaisun toimittajat	Emojulkaisun toimittajat merkitään kaikki samaan kenttään. Nimet ilmoitetaan siinä muodossa ja järjestyksessä, jossa ne on listattu alkuperäisessä julkaisussa. Mikäli kyseisen kentän tiedot sisältävät puolipisteitä (julkaisulla enemmän kuin yksi tekijä), kentän alkuun ja loppuun lisätään lainausmerkit. Koskee julkaisutyyppejä A3, B2, C2, D2, D5.
21. Kustantaja	Vapaamuotoinen tekstikenttä. Koskee julkaisutyyppejä A3, B2, C1, C2, D2, D5, E2.
22. Julkaisun kustannuspaikka	Vapaamuotoinen tekstikenttä. Julkaisun kustantajan nimen yhteydessä ilmoitettu paikkakunta tai paikkakunnat. Koskee julkaisutyyppejä A3, B2, C1, C2, D2, D5, E2.
23. Julkaisumaa	Koskee julkaisutyyppejä A-E. Tilastokeskuksen valtiot ja maat 2007 (http://www.stat.fi/meta/luokitukset/valtio/001-2007-09-21/index.html) -**luokituksen mukainen 3-numeroinen arvo**, esim. 246.

24. Julkaisun kansainvälisyys	Koskee vain julkaisutyyppejä F-I. **Koodi**: 0 - kotimainen julkaisu tai 1 - kv-julkaisu
25. DOI tunniste	Vapaamuotoinen tekstikenttä, esim.10.1038/ng1104-1133.
26. Pysyvä verkko-osoite (vapaaehtoinen)	Verkko-osoite, esim. http://dx.doi.org/10.1038/ng1104-1133.
27. Avoin saatavuus	**Koodi**: 0 = ei Open access -julkaisu tai 1 = en access -julkaisu tai 2 = Open access -julkaisu, rinnakkaistallennettu tai 9 = Ei tietoa
28. Lähdetietokannan koodi (vapaaehtoinen)	Julkaisun tunniste tai ID-numero tietokannassa, josta sen tietue on haravoitu. Jos koodeja on useampia, erotetaan ne toisistaan puolipisteellä (esim. ISI:000275364300009; PMID:20036235). Lähdetietokantoja ovat mm. ISI Web of Science, Scopus, Pubmed, ArXiv, Cab Abstracts, Arto, Fennica.
29. EVO-julkaisu (vapaaehtoinen)	**Koodi**: 0 - ei EVO-julkaisu tai 1 - EVO-julkaisu
30. Julkaisufoorumiluokka (vapaaehtoinen)	**Koodi:** 0 – ei jufo-tasoa tai 1 – ensimmäinen jufo-taso 2 – toinen jufo-taso 3- kolmas jufo-taso
31. Korkeakoulukohtainen id (vapaaehtoinen)	Vapaamuotoinen tekstikenttä.

(Ammattikorkeakoulujen tiedonkeruukäsikirja 2012, 21–35)

Liite 2

KIRJAUTUMISLOMAKKEEN CSS- JA CSS3-TYYLIOHJEET

```
body {      margin:0px;
            padding:0px;
            font-family:Verdana,Arial,Helvetica,sans-serif !important;
            line-height:22px;}

#login{
            margin-top:100px;
            margin-left:auto;
            margin-right:auto;
            text-align:center;
            width:250px;}

#login fieldset{
            border-style:solid;
            border-width:1px;
            border-color:#005a84;
            -moz-box-shadow:3px 3px 3px 1px #bababa;
            -webkit-box-shadow:3px 3px 3px 1px #bababa;
            box-shadow:3px 3px 3px 1px #bababa;}

#login .container{
            margin:15px 0px;}

#login .container img{
            margin-bottom: 10px;}

#login input{
            height:25px;
            border-style:solid;
            border-width:1px;
            border-color:#005a84;
            text-align:center;}

#login .button{
            font-size:105%;
            border-style:solid;
            border-width:1px;
            border-color:#005a84;
            background:white;
            height:30px;
            width:80px;
            color:#005a84;
            margin:5px 0px 10px 0px;}

#login input[type="submit"]:hover{
            background-color:#005a84;
            color:white;}

#varoitus{
            margin-top:20px;
            margin-left:auto;
            margin-right:auto;
            text-align:center;
            width:800px;}
```

Liite 3

UUDEN KIRJAN LISÄÄMISEEN KÄYTETTÄVÄ SIVU